# Demo_PyMDU_Atelier

June 23, 2025

## 1 Démonstration de l'utilisation de la bibliothèque pyMDU

Ce notebook a pour objectif de démontrer comment utiliser la bibliothèque pymdu

[1]:
```
import os
import sys
from pathlib import Path

import contextily as ctx
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt
import rasterio.plot
from matplotlib import rcParams
from shapely.geometry import box

%matplotlib inline
rcParams['font.family'] = 'DejaVu Sans'
```

### 1.1 Chemin de base vers l'environnement Micromamba / Conda

[2]:
```
env_dir = Path.home() / 'miniforge3' / 'envs' / 'pymdu'
# env_dir = Path.home() / 'micromamba'/'envs'/'pymdu'

if sys.platform.startswith('win'):
    # Windows
    proj_lib_path = env_dir / 'Library' / 'share' / 'proj'
    gdalwarp_exe = env_dir / 'Library' / 'bin' / 'gdalwarp.exe'
    gdal_rasterize_exe = env_dir / 'Library' / 'bin' / 'gdal_rasterize.exe'
    bin_dir = env_dir / 'Library' / 'bin'
else:
    # Linux/macOS
    proj_lib_path = env_dir / 'share' / 'proj'
    gdalwarp_exe = env_dir / 'bin' / 'gdalwarp'
    # gdal_rasterize_exe = env_dir / 'bin' / 'gdal_rasterize'
    bin_dir = env_dir / 'bin'

# Application de la configuration
os.environ['PROJ_LIB'] = str(proj_lib_path)
```

```
GDALWARP_PATH = str(gdalwarp_exe)
```

### 1.2 Chemin de base vers QGIS et ses plugins

[3]:
```
if sys.platform.startswith('win'):
    # Windows
    qgis_python = env_dir / 'Library' / 'share' / 'qgis' / 'python'
    qgis_plugins = qgis_python / 'plugins'
    user_plugins = Path.home() / 'AppData' / 'Roaming' / 'QGIS' / 'QGIS3' /␣
 ↪'profiles' / 'default' / 'python' / 'plugins'
else:
    # Linux/macOS
    qgis_python = env_dir / 'share' / 'qgis' / 'python'
    qgis_plugins = qgis_python / 'plugins'
    user_plugins = Path.home() / '.local' / 'share' / 'QGIS' / 'QGIS3' /␣
 ↪'profiles' / 'default' / 'python' / 'plugins'

# Ajout des chemins à sys.path
for path in (qgis_python, qgis_plugins, user_plugins):
    sys.path.append(str(path.resolve()))
```

### 1.3 Initialisation du dossier de simulation

[4]:
```
inputs_simulation_path = os.path.join(os.getcwd(), 'results_demo/
 ↪inputs_simulation')
os.makedirs(inputs_simulation_path, exist_ok=True)

output_path = os.path.join(os.getcwd(), 'results_demo')
os.makedirs(output_path, exist_ok=True)

output_path_urock = os.path.join(output_path, 'output_urock')
os.makedirs(output_path_urock, exist_ok=True)
```

## 2 Sélection de votre zone d'intérêt

Tracez un rectangle sur la carte ci-dessous pour délimiter la région qui vous intéresse.

Une fois la sélection effectuée, cliquez sur le rectangle et copiez le texte généré.

[5]:
```
from pymdu.commons.BasicFunctions import draw_bbox_with_folium

draw_bbox_with_folium(lat=46.160329,
                      lon=-1.151139,
                      zoom_start=13)
```

```
TEMP_PATH /var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T
```

```
Output()
```

```
<IPython.core.display.HTML object>
```

### 2.1 Chargement des données GeoJSON et calculer la bounding box

Copiez-collez le JSON ci-dessous dans la variable `geojson_dict`.

Le script suivant extrait les coordonnées du polygone, détermine les longitudes et latitudes minimales et maximales, puis construit la liste `[minx, miny, maxx, maxy]`.

[6]:
```
geojson_dict = {"type": "Feature", "properties": {}, "geometry": {"type":␣
↪"Polygon", "coordinates": [
    [[-1.155254, 46.155467], [-1.155254, 46.158503], [-1.148575, 46.158503],␣
↪[-1.148575, 46.155467],
     [-1.155254, 46.155467]]]}}
# Extraire les coordonnées du polygone
coordinates = geojson_dict['geometry']['coordinates'][0]
# Calculer les valeurs min et max
minx = min([point[0] for point in coordinates])  # Minimum des longitudes (x)
miny = min([point[1] for point in coordinates])  # Minimum des latitudes (y)
maxx = max([point[0] for point in coordinates])  # Maximum des longitudes (x)
maxy = max([point[1] for point in coordinates])  # Maximum des latitudes (y)
# Créer la liste [minx, miny, maxx, maxy]
bbox_coords = [minx, miny, maxx, maxy]
```

## 3 Collecter des données

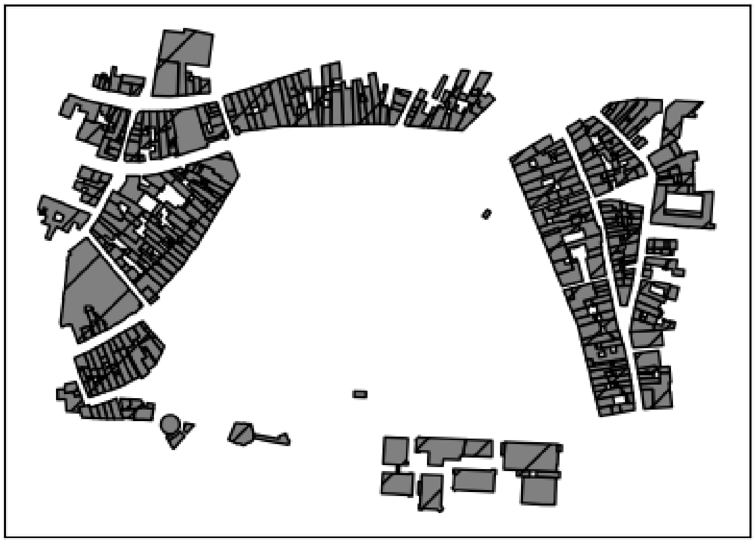
### 3.1 Bâtiments

[7]:
```
from pymdu.geometric.Building import Building

buildings = Building(output_path=inputs_simulation_path)
buildings.bbox = bbox_coords
buildings_gdf = buildings.run().to_gdf()
buildings_gdf.to_file(os.path.join(inputs_simulation_path, "buildings.shp"),␣
↪driver="ESRI Shapefile")
```

```
Index(['Service', 'Thï¿½matique', 'Producteur', 'Nom',
       'URL d'acces Geoportail', 'URL d'acces Geoplateforme',
       'Statut de licence', 'Etat de publication', 'Statut  Gï¿½oplateforme',
       'Date actualisation de la donnï¿½e', 'Remarque'],
      dtype='object')
key=> buildings
['BDTOPO_V3:batiment' 'BDTOPO_V3:batiment']
https://data.geopf.fr/wfs/ows?SERVICE=WFS&VERSION=2.0.0&REQUEST=GetCapabilities
Geo url https://data.geopf.fr/wfs/ows?SERVICE=WFS&VERSION=2.0.0
execute_ign Service WFS public de la Géoplateforme 2.0.0 WFS
typename BDTOPO_V3:batiment
```

[27]:
```python
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
buildings_gdf.plot(ax=ax, color='grey', edgecolor='k', hatch='/')
```

[27]: <Axes: >



## 3.2 Couverture du sol avec différentes couches IGN

[9]:
```python
from pymdu.geometric import Vegetation, Pedestrian, Water, LandCover

water = Water(output_path="./")
water.bbox = bbox_coords
water_gdf = water.run().to_gdf()

pedestrian = Pedestrian(output_path="./")
pedestrian.bbox = bbox_coords
pedestrian_gdf = pedestrian.run().to_gdf()

vegetation = Vegetation(output_path="./", min_area=100)
vegetation.bbox = bbox_coords
vegetation_gdf = vegetation.run().to_gdf()

landcover = LandCover(
    output_path="./",
    building_gdf=buildings_gdf,
    vegetation_gdf=vegetation_gdf,
```

```
    water_gdf=water_gdf,
    cosia_gdf=None,
    dxf_gdf=None,
    pedestrian_gdf=pedestrian_gdf,
    write_file=False,
)
landcover.bbox = bbox_coords
landcover.run()
landcover_gdf = landcover.to_gdf()
```

```
[overpass] downloading data: [timeout:25][out:json];(way["natural"="water"](46.1
81627,-1.152704,46.18699,-1.139893);relation["natural"="water"](46.181627,-
1.152704,46.18699,-1.139893);node["natural"="water"](46.181627,-
1.152704,46.18699,-1.139893);); out body geom;
```


```
{"type":"FeatureCollection","name":"OSMPythonTools","features":[{"type":"Feature
","geometry":{"type":"Polygon","coordinates":[[[-1.147657,46.183413],[-
1.147157,46.183211],[-1.146822,46.183103],[-1.146479,46.182971],[-
1.146218,46.182815],[-1.146044,46.182602],[-1.146065,46.182177],[-
1.146151,46.181804],[-1.146163,46.181986],[-1.146215,46.182176],[-
1.146338,46.182384],[-1.146628,46.182625],[-1.147196,46.182988],[-
1.147468,46.18321],[-1.147645,46.183374],[-
1.147657,46.183413]]]},"properties":{"name":"\"natural\"=\"water\"0"}},{"type":"
Feature","geometry":{"type":"Polygon","coordinates":[[[-1.151472,46.184116],[-
1.151612,46.184084],[-1.15164,46.184033],[-1.151535,46.183902],[-
1.151411,46.183899],[-1.151321,46.18396],[-1.151291,46.184006],[-
1.151321,46.184055],[-
1.151472,46.184116]]]},"properties":{"name":"\"natural\"=\"water\"1"}}]}
```


```
Index(['Service', 'Thï¿½matique', 'Producteur', 'Nom',
       'URL d'acces Geoportail', 'URL d'acces Geoplateforme',
       'Statut de licence', 'Etat de publication', 'Statut  Gï¿½oplateforme',
       'Date actualisation de la donnï¿½e', 'Remarque'],
      dtype='object')
key=> irc
['ORTHOIMAGERY.ORTHOPHOTOS.IRC' 'ORTHOIMAGERY.ORTHOPHOTOS.IRC']
https://data.geopf.fr/wms-
r/wms?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities
URL  :  /var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T/irc.tiff
```

```
ERROR 1:
_TIFFVSetField:/var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T/img.tiff: Null
count for "GeoDoubleParams" (type 12, writecount -1, passcount 1)
ERROR 1:
_TIFFVSetField:/var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T/img.tiff: Null
count for "GeoDoubleParams" (type 12, writecount -1, passcount 1)
/Users/Boris/Documents/TIPEE/pymdu/pymdu/geometric/Vegetation.py:114:
DeprecationWarning: NumPy will stop allowing conversion of out-of-bound Python
integers to integer arrays.  The conversion of -999 to uint8 will fail in the
future.
```

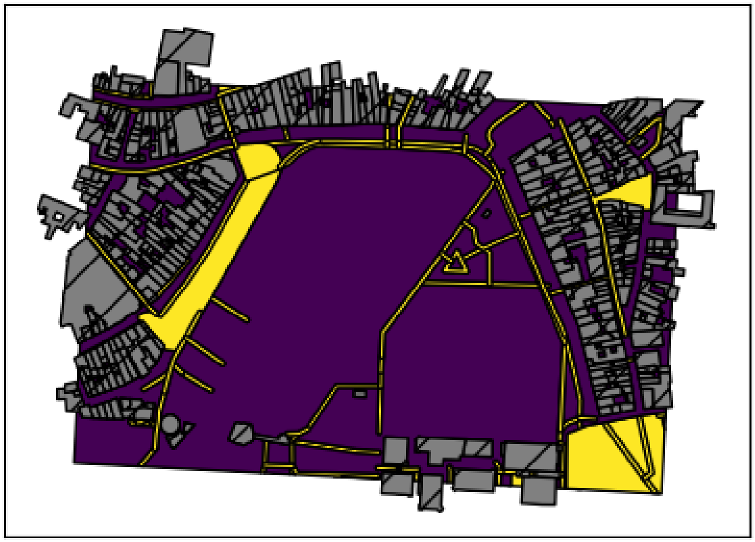
```
For the old behavior, usually:
    np.array(value).astype(dtype)
will give the desired result (the cast overflows).
  dataset_rio.data[0] = filter_raster
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/osgeo/gdal.py:311: FutureWarning: Neither gdal.UseExceptions() nor
gdal.DontUseExceptions() has been explicitly called. In GDAL 4.0, exceptions
will be enabled by default.
  warnings.warn(
/Users/Boris/Documents/TIPEE/pymdu/pymdu/geometric/Vegetation.py:174:
FutureWarning: You are adding a column named 'geometry' to a GeoDataFrame
constructed without an active geometry column. Currently, this automatically
sets the active geometry column to 'geometry' but in the future that will no
longer happen. Instead, either provide geometry to the GeoDataFrame constructor
(GeoDataFrame(… geometry=GeoSeries()) or use `set_geometry('geometry')` to
explicitly set the active geometry column.
  self.gdf["geometry"] = mes_polygons
```

0…10…20…30…40…50…60…70…80…90…

```
/Users/Boris/.local/lib/python3.11/site-packages/pandas/core/generic.py:6313:
DeprecationWarning: Overriding the CRS of a GeoDataFrame that already has CRS.
This unsafe behavior will be deprecated in future versions. Use
GeoDataFrame.set_crs method instead
  return object.__setattr__(self, name, value)
```

[10]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
landcover_gdf.plot(ax=ax, alpha=1, edgecolor="black", column="type")
buildings_gdf.plot(ax=ax, color='grey', edgecolor='k', hatch='/')
```

[10]: <Axes: >



### 3.3 Extraction et tracé cartographique des classes `COSIA` : (Couverture du Sol par Intelligence Artificielle)

Tout d'abord, téléchargez les fichiers `COSIA` correspondant à votre zone d'intérêt.

CoSIA - application pour visualiser et télécharger ses cartes de Couverture du Sol par Intelligence Artificielle.

Le lien est disponible ci-dessous.

https://cosia.ign.fr/info#export

Cette cellule importe les fichiers COSIA, calcule l'intersection avec votre zone d'intérêt, puis génère une carte où chaque polygone est coloré d'après sa classe COSIA.

#### 3.3.1 Cosia avec donnée brute GeoPackage

[11]:
```python
from pathlib import Path
from pymdu.geometric.Cosia import Cosia

directory_path = Path.home() / 'Downloads/CoSIA_D017_2021'
# directory_path = Path.home() / 'cosia/CoSIA_D017_2021'
cosia = Cosia(directory_path_cosia=directory_path)
cosia.bbox = bbox_coords
cosia_gdf = cosia.run()
table_color_cosia = cosia.table_color_cosia
cosia_gdf['color'] = [table_color_cosia[x] for x in cosia_gdf.classe]
```

```
Index(['Service', 'Thï¿½matique', 'Producteur', 'Nom',
       'URL d'acces Geoportail', 'URL d'acces Geoplateforme',
       'Statut de licence', 'Etat de publication', 'Statut  Gï¿½oplateforme',
```

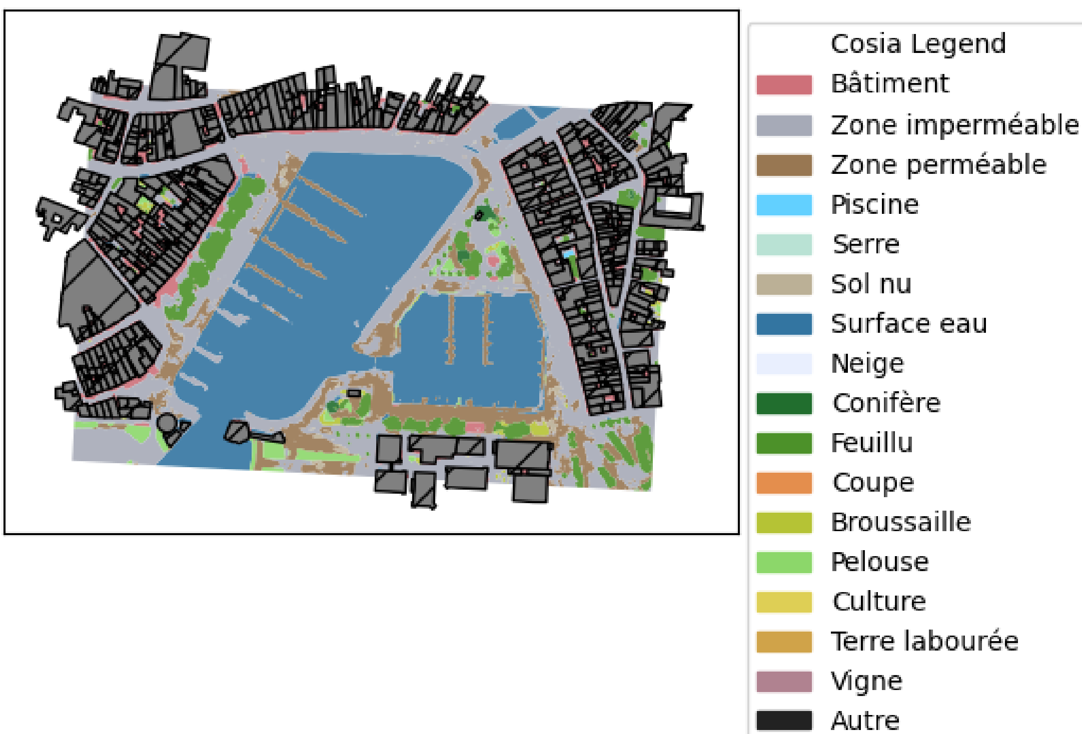
```
       'Date actualisation de la donnï¿½e', 'Remarque'],
      dtype='object')
```

```
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:198: RuntimeWarning: Field format 'character
varying(256)' not supported
  return ogr_read(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:198: RuntimeWarning: Field format 'character
varying(30)' not supported
  return ogr_read(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:198: RuntimeWarning: Field format 'character varying'
not supported
  return ogr_read(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:198: RuntimeWarning: Field format 'timestamp with time
zone' not supported
  return ogr_read(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/geopandas.py:275: UserWarning: More than one layer found in
'D017_2021_370_6580_vecto.gpkg': 'D017_2021_370_6580_vecto' (default),
'layer_styles'. Specify layer parameter to avoid this warning.
  result = read_func(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:198: RuntimeWarning: Field format 'character
varying(256)' not supported
  return ogr_read(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:198: RuntimeWarning: Field format 'character
varying(30)' not supported
  return ogr_read(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:198: RuntimeWarning: Field format 'character varying'
not supported
  return ogr_read(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:198: RuntimeWarning: Field format 'timestamp with time
zone' not supported
  return ogr_read(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/geopandas.py:275: UserWarning: More than one layer found in
'D017_2021_380_6580_vecto.gpkg': 'D017_2021_380_6580_vecto' (default),
'layer_styles'. Specify layer parameter to avoid this warning.
  result = read_func(
```

[12]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
```

```python
# Créer les patches pour chaque couleur et sa description dans la légende
patches = [
    mpatches.Patch(color=value, label=label)
    for (value, label) in zip(table_color_cosia.values(), table_color_cosia.
 ↪keys())
]

# Ajouter la légende personnalisée
plt.legend(
    handles=patches,
    loc="upper right",
    title="Cosia Legend",
    bbox_to_anchor=(1.5, 1.)
)
ax.set_xticks([])
ax.set_yticks([])
cosia_gdf.plot(ax=ax, edgecolor=None, color=cosia_gdf['color'], alpha=0.9)
buildings_gdf.plot(ax=ax, color='grey', edgecolor='k', hatch='/')
```

[12]: <Axes: >



### 3.3.2 Cosia avec donnée IGN

[13]:
```python
import rasterio
from rasterio.features import shapes
import geopandas as gpd
from shapely.geometry import shape

cosia_ign = cosia.run_ign()

with rasterio.open(cosia.path_save_tiff) as src:
    band1 = src.read(1)
    band2 = src.read(2)
    band3 = src.read(3)
    mask = band1 != src.nodata  # ignorer les valeurs nodata

    results = (
        {'properties': {'value': v}, 'geometry': s}
        for s, v in shapes(band1, mask=mask, transform=src.transform)
    )

    # Créer un GeoDataFrame à partir des résultats
    geoms = []
    values = []
    for result in results:
        geoms.append(shape(result['geometry']))
        values.append(result['properties']['value'])

cosia_gdf_ign = gpd.GeoDataFrame({'value': values, 'geometry': geoms}, crs=src.
 ↪crs)

# Afficher un aperçu
cosia_gdf_ign.head(100)
```

```
key=> cosia
['IGNF_COSIA_2021-2023_WMS']
https://data.geopf.fr/wms-
r/wms?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities
URL  :  /var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T/cosia.tiff

ERROR 1:
_TIFFVSetField:/var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T/cosia.tiff: Null
count for "GeoDoubleParams" (type 12, writecount -1, passcount 1)
```

[13]:

| | value | geometry |
|---|---|---|
| 0 | 180.0 | POLYGON ((379487.147 6570510.873, 379487.147 6… |
| 1 | 204.0 | POLYGON ((379488.147 6570510.873, 379488.147 6… |
| 2 | 192.0 | POLYGON ((379487.147 6570509.873, 379487.147 6… |
| 3 | 188.0 | POLYGON ((379532.147 6570508.873, 379532.147 6… |

```
4   200.0  POLYGON ((379487.147 6570507.873, 379487.147 6…
..    …                                              …
95  200.0  POLYGON ((379527.147 6570498.873, 379527.147 6…
96  172.0  POLYGON ((379528.147 6570498.873, 379528.147 6…
97  204.0  POLYGON ((379537.147 6570498.873, 379537.147 6…
98  204.0  POLYGON ((379585.147 6570498.873, 379585.147 6…
99  200.0  POLYGON ((379586.147 6570498.873, 379586.147 6…

[100 rows x 2 columns]
```

[14]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
cosia_gdf_ign.plot(ax=ax, edgecolor=None, column='value', alpha=0.9)
buildings_gdf.plot(ax=ax, color='grey', edgecolor='k', hatch='/')
```

[14]: <Axes: >

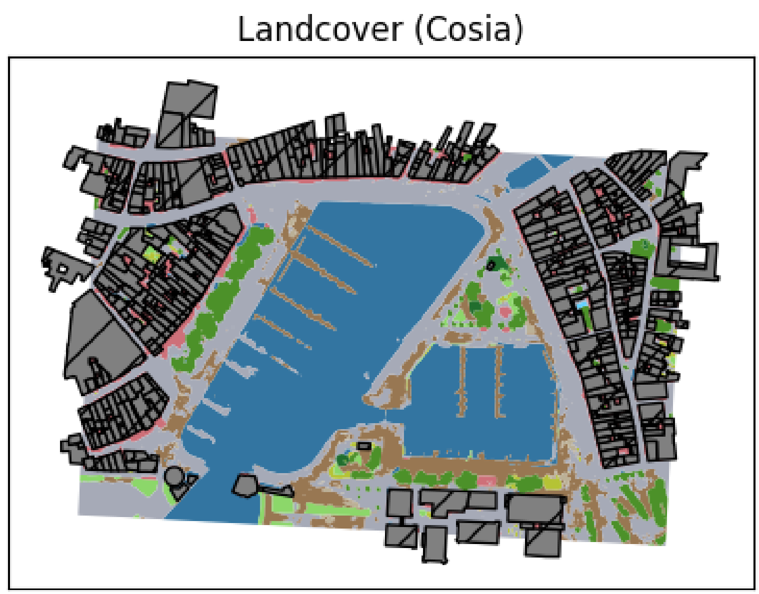
### 3.4 Couverture du sol avec différentes avec COSIA

[15]:
```
from pymdu.geometric.LandCover import LandCover

landcover = LandCover(output_path=output_path,
                      building_gdf=None,
                      vegetation_gdf=None,
                      cosia_gdf=cosia_gdf,
                      dxf_gdf=None,
```

```
                        pedestrian_gdf=None,
                        water_gdf=None)

landcover_gdf = landcover.run(keep_geom_type=True).to_gdf()
landcover.bbox = bbox_coords
landcover.to_shp(name='landcover')

landcover.create_landcover_from_cosia(os.path.join(inputs_simulation_path,␣
 ↪"landcover.tif"))
```

[16]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
plt.title('Landcover (Cosia)')
landcover.gdf.plot(ax=ax, color=landcover_gdf["color"], alpha=1)
buildings_gdf.plot(ax=ax, color='grey', edgecolor='k', hatch='/')
```

[16]:
```
<Axes: title={'center': 'Landcover (Cosia)'}>
```




### 3.5 Création de la couche DEM : (Digital Elevation Model)

Dans cette section, nous procédons à la création de la couche DEM (Digital Elevation Model), qui représente le modèle numérique de terrain pour la zone d'étude. Les données nécessaires à la construction de cette couche sont téléchargées à partir du serveur de l'IGN (Institut Géographique National), garantissant ainsi une haute précision et une couverture complète du territoire concerné.

Le DEM ne prend pas en compte les objets présents à la surface du terrain tels que les plantes et les bâtiments.

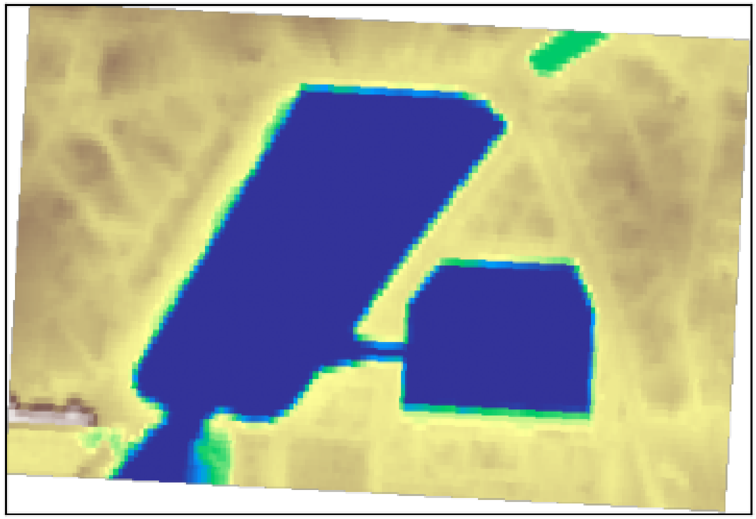
[17]:
```
from pymdu.geometric.Dem import Dem

dem = Dem(output_path=inputs_simulation_path)
dem.bbox = bbox_coords
dem.run()
```

```
Index(['Service', 'Thï¿½matique', 'Producteur', 'Nom',
       'URL d'acces Geoportail', 'URL d'acces Geoplateforme',
       'Statut de licence', 'Etat de publication', 'Statut  Gï¿½oplateforme',
       'Date actualisation de la donnï¿½e', 'Remarque'],
      dtype='object')
key=> dem
['ELEVATION.ELEVATIONGRIDCOVERAGE.HIGHRES'
 'ELEVATION.ELEVATIONGRIDCOVERAGE.HIGHRES']
https://data.geopf.fr/wms-
r/wms?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetCapabilities
URL  :  /var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T/dem.tiff
```

```
ERROR 1:
_TIFFVSetField:/var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T/dem.tiff: Null
count for "GeoDoubleParams" (type 12, writecount -1, passcount 1)
```

[17]:
```
<pymdu.geometric.Dem.Dem at 0x34dc13150>
```

[18]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
raster = rasterio.open(os.path.join(inputs_simulation_path, "DEM.tif"))
im = rasterio.plot.show(raster, ax=ax, title="Raster DEM (Digital Elevation␣
 ↪Model", cmap='terrain')
# Ajouter la barre de couleur
# fig.colorbar(im, ax=ax, orientation='vertical', label='Elevation')
plt.show()
```




### 3.6 Découpage et reprojection du DEM avec GDAL

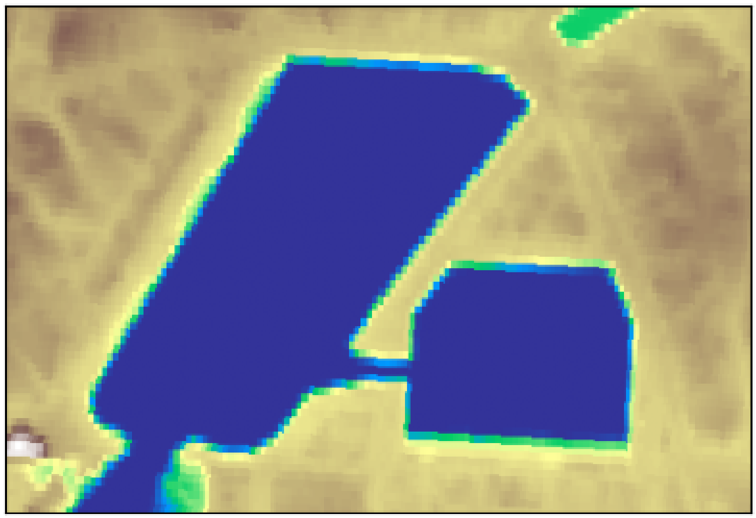
[19]:
```
import subprocess

inraster = os.path.join(inputs_simulation_path, "DEM.tif")
outraster = os.path.join(output_path, "DEM.tif")
inshape = os.path.join(inputs_simulation_path, "mask.shp")
subprocess.call([GDALWARP_PATH, inraster, outraster, '-cutline', inshape,
                 '-crop_to_cutline', '-overwrite', '-of', 'GTIFF', '-t_srs',␣
 ↪'EPSG:2154', '-tr', '1', '1', '-ot',
                 'Float32'])
```

```
Creating output file that is 453P x 309L.
Using internal nodata values (e.g. -99999) for image
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/DEM.tif.
Copying nodata values from source
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/DEM.tif
to destination /Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/DEM.tif.
Processing
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/DEM.tif
[1/1] : 0…10…20…30…40…50…60…70…80…90…100 - done.
```

[19]: 0

[20]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
```

```
raster = rasterio.open(os.path.join(output_path, "DEM.tif"))
rasterio.plot.show(raster, ax=ax, title="Raster DEM (Digital Elevation Model",␣
↪cmap='terrain')
plt.show()
```




### 3.7 Homégénéisation des rasters

Dans cette étape, nous procédons à l'homogénéisation des rasters utilisés pour les différentes couches géospatiales. Lors de la manipulation des données raster, les différences de projections peuvent entraîner des décalages spatiaux entre les couches, ce qui pourrait compromettre la précision des analyses.

Pour garantir que les résultats des simulations soient cohérents et fiables, il est essentiel de s'assurer que tous les rasters ont la même taille et les mêmes dimensions.

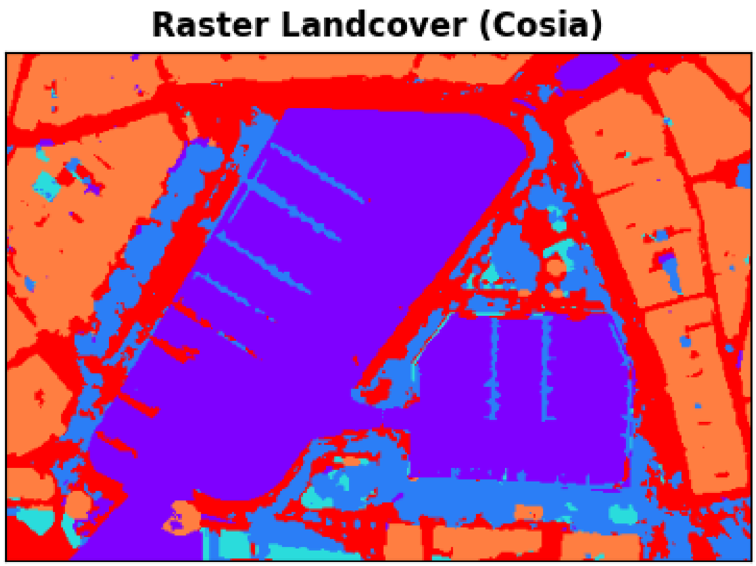
[21]:
```
from osgeo import gdal, gdalconst
from pymdu.image.geotiff import raster_file_like

gdal.AllRegister()
warp_options = gdal.WarpOptions(format='GTiff',
                                xRes=1, yRes=1,
                                outputType=gdalconst.GDT_Float32,
                                dstNodata=None,
                                dstSRS='EPSG:2154',
                                cropToCutline=True,
                                cutlineDSName=os.path.
↪join(inputs_simulation_path, 'mask.shp'),
                                cutlineLayer='mask')
```

```
gdal.Warp(destNameOrDestDS=os.path.join(output_path, 'landcover_clip.tif'),
          srcDSOrSrcDSTab=os.path.join(inputs_simulation_path, 'landcover.tif'),
          options=warp_options)

raster_file_like(src_tif=os.path.join(output_path, "landcover_clip.tif"),
                 dst_tif=os.path.join(output_path, "landcover.tif"),
                 like_path=os.path.join(output_path, "DEM.tif"),
                 remove_nan=True)
```

Pas besoin de re-découper

[21]:
```
<xarray.DataArray (band: 1, y: 309, x: 453)> Size: 560kB
array([[[2., 2., …, 2., 2.],
        [2., 2., …, 2., 2.],
        …,
        [1., 1., …, 6., 6.],
        [1., 1., …, 6., 6.]]], dtype=float32)
Coordinates:
  * band         (band) int64 8B 1
  * x            (x) float64 4kB 3.795e+05 3.795e+05 … 3.8e+05 3.8e+05
  * y            (y) float64 2kB 6.57e+06 6.57e+06 … 6.57e+06 6.57e+06
    spatial_ref  int64 8B 0
Attributes:
    long_name:      type
    name:           type
    AREA_OR_POINT:  Area
    scale_factor:   1.0
    add_offset:     0.0
```

[22]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
raster = rasterio.open(os.path.join(output_path, "landcover.tif"))
rasterio.plot.show(raster, ax=ax, title="Raster Landcover (Cosia)",␣
 ↪cmap='rainbow_r')
plt.show()
```




### 3.8 Extraction des arbres à partir de données LiDAR

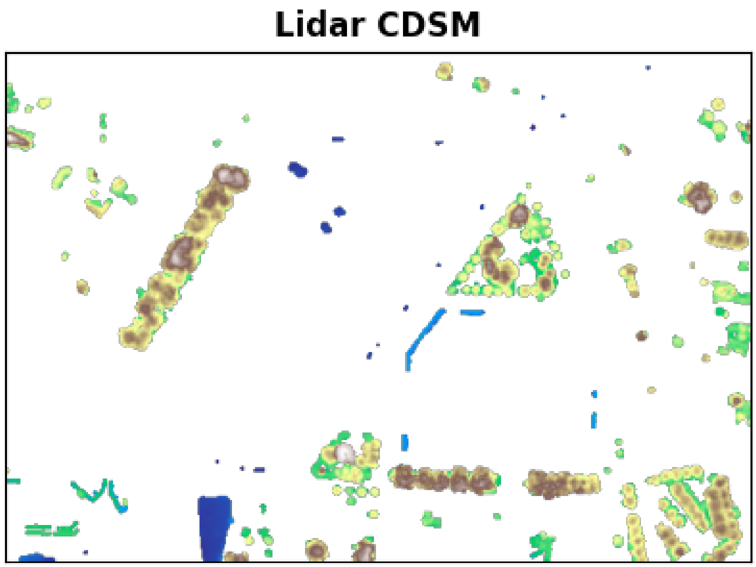
Le LiDAR (Light Detection And Ranging) est une méthode de télédétection par laser qui fournit un nuage de points 3D extrêmement précis de la surface du sol et de la végétation.

Dans cette étape, nous exploitons ces données pour détecter les arbres au sein de notre zone d'intérêt. Avec la classe `Lidar` de `pymdu` : - Nous chargeons les données LiDAR, - Appliquons la bounding box définie précédemment, - Exécutons l'algorithme de détection des arbres, - Et exportons les emplacements des arbres sous forme de shapefile.

[23]:
```
from pymdu.image.Lidar import Lidar

lidar = Lidar(output_path=inputs_simulation_path)
lidar.bbox = bbox_coords
lidar_tif = lidar.to_tif(write_out_file=True, classification_list=[3, 4, 5, 9])
```

```
<Response [200]>
```

[24]:
```
%matplotlib inline
# Lire les données et les afficher avec rasterio.plot
with lidar_tif.open() as src:
    fig, ax = plt.subplots(figsize=(5, 5))
    ax.set_xticks([])
    ax.set_yticks([])
    rasterio.plot.show(src, ax=ax, title="Lidar CDSM", cmap='terrain')
    plt.show()
```



[25]:
```
lidar_trees_gdf = lidar.run_trees()
lidar_trees_gdf.to_file(os.path.join(inputs_simulation_path, 'lidar_trees.shp'))
```

```
<Response [200]>
Downloading LAZ file from:
https://storage.sbg.cloud.ovh.net/v1/AUTH_63234f509d6048bca3c9fd7928720ca1/ppk-
lidar/FK/LHD_FXX_0379_6571_PTS_O_LAMB93_IGN69.copc.laz
Downloading LAZ file from:
https://storage.sbg.cloud.ovh.net/v1/AUTH_63234f509d6048bca3c9fd7928720ca1/ppk-
lidar/FK/LHD_FXX_0380_6571_PTS_O_LAMB93_IGN69.copc.laz
Projected BBOX (EPSG:2154): [379469.14715032035, 380001.58724372747,
6570174.188856952, 6570483.784653484]
DSM.tif saved successfully.
DTM.tif saved successfully.
CHM.tif saved successfully.
Detected 32 tree top candidates.
Extracted 32 crown polygons.
Tree crowns saved to 'tree_crowns.shp'.
Tree tops saved to 'tree_tops.shp'.
```

```
/Users/Boris/Documents/TIPEE/pymdu/pymdu/image/Lidar.py:345: UserWarning: Column
names longer than 10 characters will be truncated when saved to ESRI Shapefile.
  gdf_crowns.to_file(crown_shp)
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:723: RuntimeWarning: Normalized/laundered field name:
'tree_height' to 'tree_heigh'
  ogr_write(
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:723: RuntimeWarning: Normalized/laundered field name:
```

```
'trunk_height' to 'trunk_heig'
  ogr_write(
```

[26]:
```
%matplotlib inline
# 1. Create the bbox polygon in WGS84 (EPSG:4326)

bbox_poly = gpd.GeoSeries([box(*bbox_coords)], crs="EPSG:4326")

# 2. Convert all layers to Web Mercator (EPSG:3857)
lidar_trees_3857 = lidar_trees_gdf.to_crs(epsg=3857)
buildings_3857 = buildings_gdf.to_crs(epsg=3857)
bbox_3857 = bbox_poly.to_crs(epsg=3857)

# 3. Plot everything, including the bbox
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])

lidar_trees_3857.plot(ax=ax, color='g', alpha=1)
buildings_3857.plot(ax=ax, color='r', alpha=1)
bbox_3857.plot(ax=ax, facecolor='none', edgecolor='blue', linewidth=2)

ctx.add_basemap(ax, source=ctx.providers.Esri.WorldImagery)

plt.show()
```

# 4 Utilisation automatique du plugin UMEP de QGIS

1. Télécharger https://github.com/UMEP-dev/UMEP-processing -> renommer processing_umep
2. Coller dans le répertoire : `.local/share/QGIS/QGIS3/profiles/default/python/plugins` ou

`[...]/envs/pymdu/share/qgis/python/plugins`

## 4.1 Construction de la couche DSM : (Digital Surface Model)

Dans cette étape, nous procédons à la construction de la couche `DSM` (Digital Surface Model), qui est représentée par le fichier `DSM.tif`.

Dans le cadre du code `Solweig`, cette couche joue un rôle essentiel car elle représente la hauteur des éléments présents à la surface, tels que les bâtiments, la végétation, et autres structures. Contrairement au modèle numérique de terrain (DEM) qui ne prend en compte que la topographie du sol, le DSM inclut l'élévation des objets se trouvant au-dessus du sol.

[28]:
```
from pymdu.physics.umep.DsmModelGenerator import DsmModelGenerator

dsm = DsmModelGenerator(
    working_directory=inputs_simulation_path,
    output_filepath_dsm=os.path.join(inputs_simulation_path, "DSM.tif"),
    input_filepath_dem=os.path.join(inputs_simulation_path, "DEM.tif"),
    input_building_shp_path=os.path.join(inputs_simulation_path, "buildings.
  ↪shp"),
    input_mask_shp_path=os.path.join(inputs_simulation_path, "mask.shp")

)
dsm.run()

inraster = os.path.join(inputs_simulation_path, f"DSM.tif")
outraster = os.path.join(output_path, f"DSM.tif")
inshape = os.path.join(inputs_simulation_path, "mask.shp")

subprocess.call([GDALWARP_PATH, inraster, outraster, '-cutline', inshape,
                 '-crop_to_cutline', '-overwrite', '-of', 'GTIFF', '-t_srs',␣
  ↪'EPSG:2154', '-tr', '1', '1', '-ot',
                 'Float32'])
```

```
__init__ QGisCore
__init__ qgsApp
platform.system() Darwin
__init__ UmepCore
extent 379509.0801573259,379961.654236722,6570174.381785381,6570483.590226257
[EPSG:2154]
```

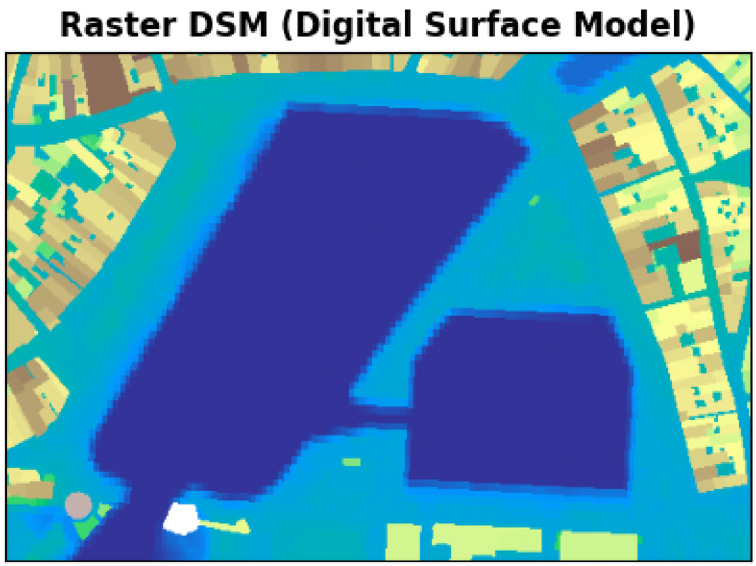
```
/Users/Boris/miniforge3/envs/pymdu/lib/python3.11/site-
packages/pyogrio/raw.py:198: RuntimeWarning: driver ESRI Shapefile does not
```

```
support open option DRIVER
  return ogr_read(
```

```
Processing UMEP umep:Spatial Data: DSM Generator
{'INPUT_DEM': '/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simu
lation/DEM.tif', 'INPUT_POLYGONLAYER': '/Users/Boris/Documents/TIPEE/pymdu/demos
/results_demo/inputs_simulation/buildings.shp', 'INPUT_FIELD': 'hauteur',
'USE_OSM': False, 'BUILDING_LEVEL': 3.1, 'EXTENT':
'379509.0801573259,379961.654236722,6570174.381785381,6570483.590226257
[EPSG:2154]', 'PIXEL_RESOLUTION': 1, 'OUTPUT_DSM': '/Users/Boris/Documents/TIPEE
/pymdu/demos/results_demo/inputs_simulation/DSM.tif'}
Processing UMEP EXIT umep:Spatial Data: DSM Generator
Creating output file that is 453P x 309L.
Using internal nodata values (e.g. -9999) for image
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/DSM.tif.
Copying nodata values from source
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/DSM.tif
to destination /Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/DSM.tif.
Processing
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/DSM.tif
[1/1] : 0…10…20…30…40…50…60…70…80…90…100 - done.
```

[28]: 0

[29]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
raster = rasterio.open(os.path.join(output_path, "DSM.tif"))
img = rasterio.plot.show(raster, ax=ax, title="Raster DSM (Digital Surface␣
  ↪Model)", cmap='terrain')
plt.show()
```




j## Construction de la couche CDSM et TDSM

Dans cette étape, nous procédons à la construction des couches CDSM (Canopy Digital Surface Model) et TDSM (Tree Digital Surface Model), qui sont essentielles pour les simulations dans le cadre du code Solweig. La couche CDSM représente un modèle numérique de surface spécifique à la canopée urbaine, c'est-à-dire les éléments au-dessus du sol qui ne sont pas des bâtiments, principalement des haies et le tronc des arbres.

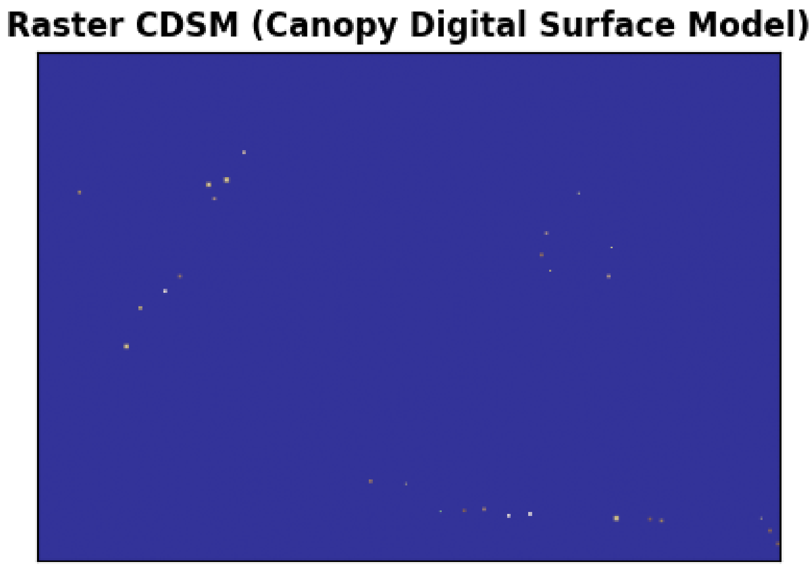
De son côté, la couche TDSM est dédiée à la représentation des arbres. Elle modélise l'élévation des arbres, ce qui permet d'analyser leur contribution à la régulation thermique et à la réduction des îlots de chaleur en milieu urbain.

[30]:
```
from pymdu.physics.umep.SurfaceModelGenerator import SurfaceModelGenerator

trees_path = os.path.join(inputs_simulation_path, 'lidar_trees.shp')

surface_model = SurfaceModelGenerator(
    working_directory=inputs_simulation_path,
    input_filepath_dsm=os.path.join(output_path, "DSM.tif"),
    input_filepath_dem=os.path.join(output_path, "DEM.tif"),
    input_filepath_tree_shp=trees_path,
    output_filepath_cdsm=os.path.join(inputs_simulation_path, "CDSM.tif"),
    output_filepath_tdsm=os.path.join(inputs_simulation_path, "TDSM.tif")
)
surface_model.run()
list_files = ['CDSM', 'TDSM']

for file in list_files:
    inraster = os.path.join(inputs_simulation_path, f"{file}.tif")
```

```
    outraster = os.path.join(output_path, f"{file}.tif")
    inshape = os.path.join(inputs_simulation_path, "mask.shp")

    subprocess.call([GDALWARP_PATH, inraster, outraster, '-cutline', inshape,
                     '-crop_to_cutline', '-overwrite', '-of', 'GTIFF',␣
  ↪'-t_srs', 'EPSG:2154', '-tr', '1', '1', '-ot',
                     'Float32'])
```

```
__init__ QGisCore
__init__ qgsApp
platform.system() Darwin
__init__ UmepCore
Processing UMEP umep:Spatial Data: Tree Generator
{'INPUT_POINTLAYER': '/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inpu
ts_simulation/lidar_trees.shp', 'TREE_TYPE': 'type', 'TOT_HEIGHT': 'height',
'TRUNK_HEIGHT': 'trunk zone', 'DIA': 'diameter', 'INPUT_BUILD': None,
'INPUT_DSM': '/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/DSM.tif',
'INPUT_DEM': '/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/DEM.tif',
'INPUT_CDSM': None, 'INPUT_TDSM': None, 'CDSM_GRID_OUT': '/Users/Boris/Documents
/TIPEE/pymdu/demos/results_demo/inputs_simulation/CDSM.tif', 'TDSM_GRID_OUT': '/
Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/TDSM.tif'
}
Processing UMEP EXIT umep:Spatial Data: Tree Generator
Creating output file that is 453P x 309L.
Processing
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/CDSM.tif
[1/1] : 0…10…20…30…40…50…60…70…80…90…100 - done.
Creating output file that is 453P x 309L.
Processing
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/TDSM.tif
[1/1] : 0…10…20…30…40…50…60…70…80…90…100 - done.
```

[31]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
raster = rasterio.open(os.path.join(inputs_simulation_path, "CDSM.tif"))
rasterio.plot.show(raster, ax=ax, title="Raster CDSM (Canopy Digital Surface␣
  ↪Model)", cmap='terrain')
plt.show()
```




### 4.2 Construction de la couche HEIGHT et ASPECT

Dans cette étape, nous procédons à la construction des couches `HEIGHT` et `ASPECT`, qui jouent un rôle crucial dans les simulations climatiques effectuées avec le code `Solweig`.

La couche `HEIGHT` représente la hauteur des structures urbaines, telles que les bâtiments, par rapport au niveau du sol. Cette information est fondamentale pour évaluer l'impact des différentes hauteurs sur la distribution des ombres, la répartition des rayonnements solaires, et, par conséquent, sur la température ressentie dans l'environnement urbain.

La couche `ASPECT`, indique l'orientation des pentes et des surfaces par rapport aux points cardinaux. Cette orientation est essentielle pour comprendre comment les surfaces captent ou réfléchissent la lumière du soleil tout au long de la journée, influençant directement la distribution des températures et des conditions microclimatiques au sein du quartier.

[32]:
```
from pymdu.physics.umep.HeightAspectModelGenerator import␣
↪HeightAspectModelGenerator

height_aspect_model = HeightAspectModelGenerator(
    working_directory=inputs_simulation_path,
    output_filepath_height=os.path.join(inputs_simulation_path, "HEIGHT.tif"),
    output_filepath_aspect=os.path.join(inputs_simulation_path, "ASPECT.tif"),
    input_filepath_dsm=os.path.join(inputs_simulation_path, "DSM.tif"),

)
height_aspect_model.run()

list_files = ['HEIGHT', 'ASPECT']
```

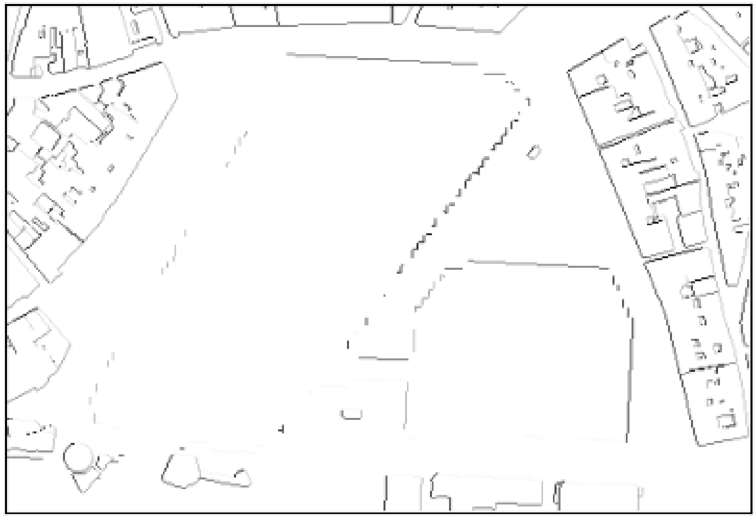
```
for file in list_files:
    inraster = os.path.join(inputs_simulation_path, f"{file}.tif")
    outraster = os.path.join(output_path, f"{file}.tif")
    inshape = os.path.join(inputs_simulation_path, "mask.shp")

    subprocess.call([GDALWARP_PATH, inraster, outraster, '-cutline', inshape,
                     '-crop_to_cutline', '-overwrite', '-of', 'GTIFF',␣
 ↪'-t_srs', 'EPSG:2154', '-tr', '1', '1', '-ot',
                     'Float32'])
```

```
__init__ QGisCore
__init__ qgsApp
platform.system() Darwin
__init__ UmepCore
Processing UMEP umep:Urban Geometry: Wall Height and Aspect
{'INPUT': '/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulati
on/DSM.tif', 'INPUT_LIMIT': 3, 'OUTPUT_HEIGHT': '/Users/Boris/Documents/TIPEE/py
mdu/demos/results_demo/inputs_simulation/HEIGHT.tif', 'OUTPUT_ASPECT': '/Users/B
oris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/ASPECT.tif'}
Processing UMEP EXIT umep:Urban Geometry: Wall Height and Aspect
Creating output file that is 453P x 309L.
Processing /Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulati
on/HEIGHT.tif [1/1] : 0…10…20…30…40…50…60…70…80…90…100 -
done.
Creating output file that is 453P x 309L.
Processing /Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulati
on/ASPECT.tif [1/1] : 0…10…20…30…40…50…60…70…80…90…100 -
done.
```

[35]:

```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
raster = rasterio.open(os.path.join(output_path, "ASPECT.tif"))
rasterio.plot.show(raster, ax=ax, title="Raster ASPECT", cmap='binary')
plt.show()
```

Raster ASPECT



[34]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
raster = rasterio.open(os.path.join(output_path, "HEIGHT.tif"))
rasterio.plot.show(raster, ax=ax, title="Raster HEIGHT", cmap='binary')
plt.show()
```

Raster HEIGHT

### 4.3 Construction de la couche `SkyViewFactor`

Dans cette étape, nous procédons à la construction de la couche `SkyViewFactor`, qui est une composante essentielle pour les simulations climatiques dans le cadre du code `Solweig`. Le Sky View Factor (SVF) est un indicateur qui mesure la proportion du ciel visible depuis un point donné au sol. Il est particulièrement important dans les environnements urbains denses, où les bâtiments et autres structures peuvent obstruer la vue du ciel, réduisant ainsi l'exposition au rayonnement solaire direct et affectant la température ressentie.

`Solweig` se distingue par sa capacité à utiliser des Sky View Factors directionnels, ce qui signifie qu'il prend en compte la visibilité du ciel dans différentes directions (nord, sud, est, ouest) pour chaque point de la zone étudiée. Cette approche directionnelle permet une modélisation plus fine des interactions entre les bâtiments, les ombres, et le climat urbain.

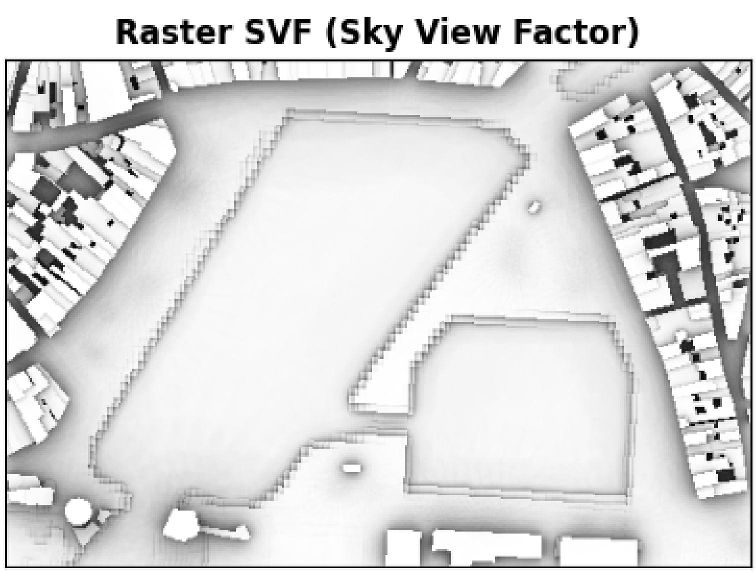
[36]:
```python
from pymdu.physics.umep.SVFModelGenerator import SVFModelGenerator

svf_model = SVFModelGenerator(
    working_directory=output_path,
    input_filepath_tdsm=os.path.join(inputs_simulation_path, "TDSM.tif"),
    input_filepath_cdsm=os.path.join(inputs_simulation_path, "CDSM.tif"),
    input_filepath_dsm=os.path.join(inputs_simulation_path, "DSM.tif"),
    ouptut_filepath_svf=os.path.join(inputs_simulation_path, "SVF.tif"),
)
svf_model.run()

inraster = os.path.join(inputs_simulation_path, "SVF.tif")
outraster = os.path.join(output_path, "SVF_clip.tif")
inshape = os.path.join(inputs_simulation_path, "mask.shp")

subprocess.call([GDALWARP_PATH, inraster, outraster, '-cutline', inshape,
                 '-crop_to_cutline', '-overwrite', '-of', 'GTIFF', '-t_srs',␣
 ↪'EPSG:2154', '-tr', '1', '1', '-ot',
                 'Float32'])
```

```
__init__ QGisCore
__init__ qgsApp
platform.system() Darwin
__init__ UmepCore
Processing UMEP umep:Urban Geometry: Sky View Factor
{'INPUT_DSM': '/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simu
lation/DSM.tif', 'INPUT_CDSM': '/Users/Boris/Documents/TIPEE/pymdu/demos/results
_demo/inputs_simulation/CDSM.tif', 'TRANS_VEG': 3, 'INPUT_TDSM': '/Users/Boris/D
ocuments/TIPEE/pymdu/demos/results_demo/inputs_simulation/TDSM.tif',
'INPUT_THEIGHT': 25.0, 'ANISO': True, 'OUTPUT_DIR':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo', 'OUTPUT_FILE': '/Users/
Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/SVF.tif'}
Processing UMEP EXIT umep:Urban Geometry: Sky View Factor
Creating output file that is 453P x 309L.
Using internal nodata values (e.g. -9999) for image
```

```
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/SVF.tif.
Copying nodata values from source
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/SVF.tif
to destination
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/SVF_clip.tif.
Processing
/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/inputs_simulation/SVF.tif
[1/1] : 0…10…20…30…40…50…60…70…80…90…100 - done.
```

[36]: 0

[37]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
raster = rasterio.open(os.path.join(output_path, "SVF_clip.tif"))
rasterio.plot.show(raster, ax=ax, title="Raster SVF (Sky View Factor)",␣
  ↪cmap='gray')
plt.show()
```




## 4.4 Météo

[38]:
```
from pymdu.meteo.Meteo import Meteo

meteo_file = Meteo(output_path=r"./")
meteo_file.bbox = bbox_coords
meteo_file.run(
    begin='2018-06-30 00:00:00',
```

```
    end='2018-06-30 23:00:00'
)
```

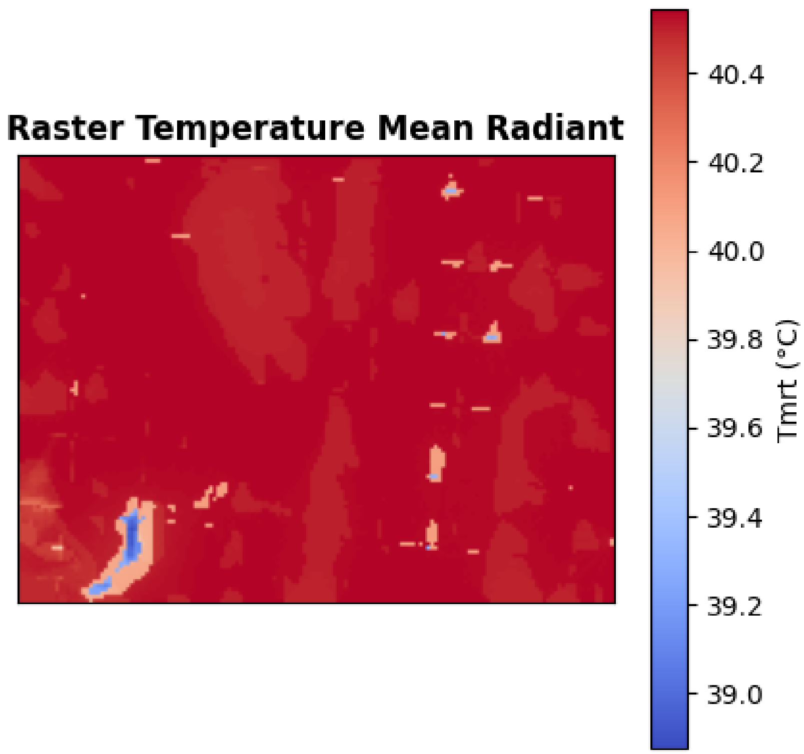
### 4.5 Calcul de la température moyenne radiante

[39]:
```
from pymdu.physics.umep.Solweig import Solweig

d = Solweig(meteo_path='FRA_AC_La.Rochelle.073150_TMYx.2004-2018.txt',
            output_dir=output_path,
            working_directory=output_path,
            input_filepath_landcover=os.path.join(output_path, "landcover.tif"),
            input_filepath_dsm=os.path.join(output_path, "DSM.tif"),
            input_filepath_dem=os.path.join(output_path, "DEM.tif"),
            input_filepath_cdsm=os.path.join(output_path, "CDSM.tif"),
            input_filepath_tdsm=os.path.join(output_path, "TDSM.tif"),
            input_filepath_height=os.path.join(output_path, "HEIGHT.tif"),
            input_filepath_aspect=os.path.join(output_path, "ASPECT.tif"),
            input_filepath_shadowmats_npz=os.path.join(output_path, "shadowmats.
↪npz"),
            input_filepath_svf_zip=os.path.join(output_path, "svfs.zip"))
d.run()
```

```
__init__ QGisCore
__init__ qgsApp
platform.system() Darwin
__init__ UmepCore
Processing UMEP umep:Outdoor Thermal Comfort: SOLWEIG
{'INPUT_DSM': '/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/DSM.tif',
'INPUT_SVF': '/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/svfs.zip',
'INPUT_HEIGHT':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/HEIGHT.tif',
'INPUT_ASPECT':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/ASPECT.tif',
'INPUT_CDSM': '/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/CDSM.tif',
'TRANS_VEG': 3, 'INPUT_TDSM':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/TDSM.tif',
'INPUT_THEIGHT': 25, 'INPUT_LC':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/landcover.tif',
'USE_LC_BUILD': False, 'INPUT_DEM':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/DSM.tif', 'SAVE_BUILD':
False, 'INPUT_ANISO':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/shadowmats.npz',
'ALBEDO_WALLS': 0.2, 'ALBEDO_GROUND': 0.15, 'EMIS_WALLS': 0.9, 'EMIS_GROUND':
0.95, 'ABS_S': 0.7, 'ABS_L': 0.95, 'POSTURE': 0.5, 'CYL': True, 'INPUTMET':
'FRA_AC_La.Rochelle.073150_TMYx.2004-2018.txt', 'ONLYGLOBAL': False, 'UTC': 0,
'POI_FILE': None, 'POI_FIELD': '', 'AGE': 35, 'ACTIVITY': 80, 'CLO': 0.9,
'WEIGHT': 75, 'HEIGHT': 180, 'SEX': 0, 'SENSOR_HEIGHT': 10, 'OUTPUT_TMRT': True,
```

```
'OUTPUT_KDOWN': False, 'OUTPUT_KUP': False, 'OUTPUT_LDOWN': False, 'OUTPUT_LUP':
False, 'OUTPUT_SH': True, 'OUTPUT_TREEPLANTER': False, 'OUTPUT_DIR':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo'}
Processing UMEP EXIT umep:Outdoor Thermal Comfort: SOLWEIG
```

[74]:
```
%matplotlib inline
fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
raster = rasterio.open(os.path.join(output_path, "Tmrt_average.tif"))
img = rasterio.plot.show(raster, ax=ax, title="Raster Temperature Mean␣
↪Radiant", cmap='coolwarm')
plt.colorbar(img.get_images()[0], ax=ax, label='Tmrt (°C)')
plt.show()
```




### 4.6 Lancement UROCK : (Urban Wind Field)

[40]:
```
import geopandas as gpd
from pymdu.geometric.UrockFiles import UrockFiles

batiments = gpd.read_file(os.path.join(inputs_simulation_path, 'buildings.shp'))
arbres = gpd.read_file(os.path.join(inputs_simulation_path, 'lidar_trees.shp'))
```

```python
urock = UrockFiles(output_path, batiments, arbres)
urock.generate_urock_buildings().to_file(os.path.join(output_path,␣
↪'batiments_urock.shp'), driver="ESRI Shapefile")
urock.generate_urock_trees().to_file(os.path.join(output_path, 'arbres_urock.
↪shp'), driver="ESRI Shapefile")
```

[41]:
```python
from pymdu.physics.umep.UmepCore import UmepCore

for direction in range(50, 55, 10):
    options_umep_urock = {
        'BUILDINGS': os.path.join(output_path, 'batiments_urock.shp'),
        'HEIGHT_FIELD_BUILD': 'hauteur',
        'VEGETATION': os.path.join(output_path, 'arbres_urock.shp'),
        'VEGETATION_CROWN_TOP_HEIGHT': 'MAX_HEIGHT',
        'VEGETATION_CROWN_BASE_HEIGHT': 'MIN_HEIGHT',
        'ATTENUATION_FIELD': 'ATTENUATIO',
        'INPUT_PROFILE_FILE': '',
        'INPUT_PROFILE_TYPE': 0,
        'INPUT_WIND_HEIGHT': 10,
        'INPUT_WIND_SPEED': 1,
        'INPUT_WIND_DIRECTION': direction,
        'RASTER_OUTPUT': None,
        'HORIZONTAL_RESOLUTION': 1,
        'VERTICAL_RESOLUTION': 10,
        'WIND_HEIGHT': '2',
        'UROCK_OUTPUT': output_path_urock,
        'OUTPUT_FILENAME': f'output_{direction}',
        'SAVE_RASTER': False,
        'SAVE_VECTOR': False,
        'SAVE_NETCDF': True,
        'LOAD_OUTPUT': True
    }

    umep_core = UmepCore(output_dir=output_path_urock)

    umep_core.run_processing(
        name="umep:Urban Wind Field: URock",
        options=options_umep_urock
    )
```

```
__init__ QGisCore
__init__ qgsApp
platform.system() Darwin
__init__ UmepCore
Processing UMEP umep:Urban Wind Field: URock
{'BUILDINGS':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/batiments_urock.shp',
```

```
'HEIGHT_FIELD_BUILD': 'hauteur', 'VEGETATION':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/arbres_urock.shp',
'VEGETATION_CROWN_TOP_HEIGHT': 'MAX_HEIGHT', 'VEGETATION_CROWN_BASE_HEIGHT':
'MIN_HEIGHT', 'ATTENUATION_FIELD': 'ATTENUATIO', 'INPUT_PROFILE_FILE': '',
'INPUT_PROFILE_TYPE': 0, 'INPUT_WIND_HEIGHT': 10, 'INPUT_WIND_SPEED': 1,
'INPUT_WIND_DIRECTION': 50, 'RASTER_OUTPUT': None, 'HORIZONTAL_RESOLUTION': 1,
'VERTICAL_RESOLUTION': 10, 'WIND_HEIGHT': '2', 'UROCK_OUTPUT':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/output_urock',
'OUTPUT_FILENAME': 'output_50', 'SAVE_RASTER': False, 'SAVE_VECTOR': False,
'SAVE_NETCDF': True, 'LOAD_OUTPUT': True}
Connecting to database
->/var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T/myDbH21750670616_7470589
/Users/Boris/miniforge3/envs/pymdu/share/qgis/python/plugins/processing_umep/fun
ctions/URock/h2gis-standalone/h2gis-dist-2.2.3.jar
100 - done.
Connected!


SLF4J(W): No SLF4J providers were found.
SLF4J(W): Defaulting to no-operation (NOP) logger implementation
SLF4J(W): See https://www.slf4j.org/codes.html#noProviders for further details.

Spatial functions added!

Load input data
Load table 'build_pre_srid_20250623112338'
Load table 'veg_pre_srid_20250623112338'
Creates blocks and stacked blocks
Rotates geometries from 50.0 degrees
Identify block base height and block cavity base
Calculates obstacle properties
Calculates zone properties
Initializes upwind facades
Update upwind facades base height
Initializes downwind facades
Calculates study area properties
Roughness zone properties are:

                                 - z0: 1.0471534870438226
                                 - d: 3.9919850322125745
                                 - Hr: 14.138873172227598
                                 - H_ob_max: 30.0
                                 - lambda_f: 0.07406201854195162
Rotates geometries from -50.0 degrees
Creates displacement zones
Creates cavity and wake zones
Creates street canyon zones
Creates rooftop zones (perpendicular and corner)
Creates built-up and open vegetation zones
```

```
Creates the grid of points
Affects each grid point to a building Rockle zone and calculates needed
          variables for 3D wind speed
Affects each grid point to a vegetation Rockle zone and calculates
          needed variables for 3D wind speed
Remove some of the Röckle zone points
Creates backward zones
Calculates the 3D wind speed factor value for each point of each BUILDING zone
Calculates the 3D wind speed factor value for each point of each VEGETATION zone
Deals with superimposition (keeps only 1 value per 3D point)
Deals with superimposition (keeps only 1 value per 3D point)
Identify upstreamer points in TEMPO_WEIGHTING table
Identify upstreamer points in TEMPO_PRIORITIES table
Identify upstreamer points in TEMPO_PRIORITIES_WEIGHT table
Deals with superimposition (keeps only 1 value per 3D point)
Identify upstreamer points in TEMPO_WEIGHTING table
Identify upstreamer points in TEMPO_PRIORITIES table
Identify upstreamer points in TEMPO_PRIORITIES_WEIGHT table
Identify grid points intersecting buildings
Set the initial 3D wind speed field
Time spent for wind speed initialization: 207.07002806663513 s
Shape: (905, 1079, 6) - Nb cells: 5858970
Start to apply the wind solver
Iteration 1 (max 500)
   eps = 0.771604 >= 0.0001
Iteration 2 (max 500)
   eps = 0.384589 >= 0.0001
Iteration 3 (max 500)
   eps = 0.284751 >= 0.0001
Iteration 4 (max 500)
   eps = 0.234257 >= 0.0001
Iteration 5 (max 500)
   eps = 0.19738 >= 0.0001
Iteration 6 (max 500)
   eps = 0.170188 >= 0.0001
Iteration 7 (max 500)
   eps = 0.147963 >= 0.0001
Iteration 8 (max 500)
   eps = 0.129967 >= 0.0001
Iteration 9 (max 500)
   eps = 0.115727 >= 0.0001
Iteration 10 (max 500)
   eps = 0.105029 >= 0.0001
Iteration 11 (max 500)
   eps = 0.096036 >= 0.0001
Iteration 12 (max 500)
   eps = 0.088804 >= 0.0001
Iteration 13 (max 500)
```

```
   eps = 0.0831 >= 0.0001
Iteration 14 (max 500)
   eps = 0.078355 >= 0.0001
Iteration 15 (max 500)
   eps = 0.074625 >= 0.0001
Iteration 16 (max 500)
   eps = 0.071458 >= 0.0001
Iteration 17 (max 500)
   eps = 0.069077 >= 0.0001
Iteration 18 (max 500)
   eps = 0.06662 >= 0.0001
Iteration 19 (max 500)
   eps = 0.064585 >= 0.0001
Iteration 20 (max 500)
   eps = 0.062863 >= 0.0001
Iteration 21 (max 500)
   eps = 0.061436 >= 0.0001
Iteration 22 (max 500)
   eps = 0.060353 >= 0.0001
Iteration 23 (max 500)
   eps = 0.059275 >= 0.0001
Iteration 24 (max 500)
   eps = 0.058037 >= 0.0001
Iteration 25 (max 500)
   eps = 0.057018 >= 0.0001
Iteration 26 (max 500)
   eps = 0.05609 >= 0.0001
Iteration 27 (max 500)
   eps = 0.055302 >= 0.0001
Iteration 28 (max 500)
   eps = 0.054608 >= 0.0001
Iteration 29 (max 500)
   eps = 0.054111 >= 0.0001
Iteration 30 (max 500)
   eps = 0.053586 >= 0.0001
Iteration 31 (max 500)
   eps = 0.053091 >= 0.0001
Iteration 32 (max 500)
   eps = 0.052647 >= 0.0001
Iteration 33 (max 500)
   eps = 0.052267 >= 0.0001
Iteration 34 (max 500)
   eps = 0.051775 >= 0.0001
Iteration 35 (max 500)
   eps = 0.05135 >= 0.0001
Iteration 36 (max 500)
   eps = 0.050932 >= 0.0001
Iteration 37 (max 500)
```

```
   eps = 0.050644 >= 0.0001
Iteration 38 (max 500)
   eps = 0.050477 >= 0.0001
Iteration 39 (max 500)
   eps = 0.050078 >= 0.0001
Iteration 40 (max 500)
   eps = 0.049675 >= 0.0001
Iteration 41 (max 500)
   eps = 0.049324 >= 0.0001
Iteration 42 (max 500)
   eps = 0.049016 >= 0.0001
Iteration 43 (max 500)
   eps = 0.048816 >= 0.0001
Iteration 44 (max 500)
   eps = 0.048691 >= 0.0001
Iteration 45 (max 500)
   eps = 0.048563 >= 0.0001
Iteration 46 (max 500)
   eps = 0.048404 >= 0.0001
Iteration 47 (max 500)
   eps = 0.048269 >= 0.0001
Iteration 48 (max 500)
   eps = 0.048222 >= 0.0001
Iteration 49 (max 500)
   eps = 0.048149 >= 0.0001
Iteration 50 (max 500)
   eps = 0.047997 >= 0.0001
Iteration 51 (max 500)
   eps = 0.047862 >= 0.0001
Iteration 52 (max 500)
   eps = 0.04783 >= 0.0001
Iteration 53 (max 500)
   eps = 0.047808 >= 0.0001
Iteration 54 (max 500)
   eps = 0.04783 >= 0.0001
Iteration 55 (max 500)
   eps = 0.047864 >= 0.0001
Iteration 56 (max 500)
   eps = 0.047934 >= 0.0001
Iteration 57 (max 500)
   eps = 0.047916 >= 0.0001
Iteration 58 (max 500)
   eps = 0.047954 >= 0.0001
Iteration 59 (max 500)
   eps = 0.048025 >= 0.0001
Iteration 60 (max 500)
   eps = 0.048146 >= 0.0001
Iteration 61 (max 500)
```

```
   eps = 0.048248 >= 0.0001
Iteration 62 (max 500)
   eps = 0.04841 >= 0.0001
Iteration 63 (max 500)
   eps = 0.048502 >= 0.0001
Iteration 64 (max 500)
   eps = 0.048586 >= 0.0001
Iteration 65 (max 500)
   eps = 0.048588 >= 0.0001
Iteration 66 (max 500)
   eps = 0.048574 >= 0.0001
Iteration 67 (max 500)
   eps = 0.048636 >= 0.0001
Iteration 68 (max 500)
   eps = 0.048499 >= 0.0001
Iteration 69 (max 500)
   eps = 0.048267 >= 0.0001
Iteration 70 (max 500)
   eps = 0.048101 >= 0.0001
Iteration 71 (max 500)
   eps = 0.047847 >= 0.0001
Iteration 72 (max 500)
   eps = 0.047637 >= 0.0001
Iteration 73 (max 500)
   eps = 0.047534 >= 0.0001
Iteration 74 (max 500)
   eps = 0.047415 >= 0.0001
Iteration 75 (max 500)
   eps = 0.047183 >= 0.0001
Iteration 76 (max 500)
   eps = 0.046985 >= 0.0001
Iteration 77 (max 500)
   eps = 0.04676 >= 0.0001
Iteration 78 (max 500)
   eps = 0.046618 >= 0.0001
Iteration 79 (max 500)
   eps = 0.046358 >= 0.0001
Iteration 80 (max 500)
   eps = 0.046155 >= 0.0001
Iteration 81 (max 500)
   eps = 0.046017 >= 0.0001
Iteration 82 (max 500)
   eps = 0.04577 >= 0.0001
Iteration 83 (max 500)
   eps = 0.045511 >= 0.0001
Iteration 84 (max 500)
   eps = 0.045298 >= 0.0001
Iteration 85 (max 500)
```

```
   eps = 0.045086 >= 0.0001
Iteration 86 (max 500)
   eps = 0.044867 >= 0.0001
Iteration 87 (max 500)
   eps = 0.044674 >= 0.0001
Iteration 88 (max 500)
   eps = 0.044451 >= 0.0001
Iteration 89 (max 500)
   eps = 0.044175 >= 0.0001
Iteration 90 (max 500)
   eps = 0.043943 >= 0.0001
Iteration 91 (max 500)
   eps = 0.043665 >= 0.0001
Iteration 92 (max 500)
   eps = 0.043479 >= 0.0001
Iteration 93 (max 500)
   eps = 0.043213 >= 0.0001
Iteration 94 (max 500)
   eps = 0.042871 >= 0.0001
Iteration 95 (max 500)
   eps = 0.042585 >= 0.0001
Iteration 96 (max 500)
   eps = 0.042342 >= 0.0001
Iteration 97 (max 500)
   eps = 0.042068 >= 0.0001
Iteration 98 (max 500)
   eps = 0.041794 >= 0.0001
Iteration 99 (max 500)
   eps = 0.041458 >= 0.0001
Iteration 100 (max 500)
   eps = 0.04123 >= 0.0001
Iteration 101 (max 500)
   eps = 0.040902 >= 0.0001
Iteration 102 (max 500)
   eps = 0.040545 >= 0.0001
Iteration 103 (max 500)
   eps = 0.040217 >= 0.0001
Iteration 104 (max 500)
   eps = 0.039977 >= 0.0001
Iteration 105 (max 500)
   eps = 0.039679 >= 0.0001
Iteration 106 (max 500)
   eps = 0.039417 >= 0.0001
Iteration 107 (max 500)
   eps = 0.039112 >= 0.0001
Iteration 108 (max 500)
   eps = 0.038859 >= 0.0001
Iteration 109 (max 500)
```

```
   eps = 0.038565 >= 0.0001
Iteration 110 (max 500)
   eps = 0.038307 >= 0.0001
Iteration 111 (max 500)
   eps = 0.038019 >= 0.0001
Iteration 112 (max 500)
   eps = 0.037821 >= 0.0001
Iteration 113 (max 500)
   eps = 0.037625 >= 0.0001
Iteration 114 (max 500)
   eps = 0.037358 >= 0.0001
Iteration 115 (max 500)
   eps = 0.037126 >= 0.0001
Iteration 116 (max 500)
   eps = 0.036903 >= 0.0001
Iteration 117 (max 500)
   eps = 0.036682 >= 0.0001
Iteration 118 (max 500)
   eps = 0.036431 >= 0.0001
Iteration 119 (max 500)
   eps = 0.036251 >= 0.0001
Iteration 120 (max 500)
   eps = 0.036068 >= 0.0001
Iteration 121 (max 500)
   eps = 0.035909 >= 0.0001
Iteration 122 (max 500)
   eps = 0.035716 >= 0.0001
Iteration 123 (max 500)
   eps = 0.035506 >= 0.0001
Iteration 124 (max 500)
   eps = 0.035312 >= 0.0001
Iteration 125 (max 500)
   eps = 0.035144 >= 0.0001
Iteration 126 (max 500)
   eps = 0.034985 >= 0.0001
Iteration 127 (max 500)
   eps = 0.034823 >= 0.0001
Iteration 128 (max 500)
   eps = 0.034609 >= 0.0001
Iteration 129 (max 500)
   eps = 0.03439 >= 0.0001
Iteration 130 (max 500)
   eps = 0.034171 >= 0.0001
Iteration 131 (max 500)
   eps = 0.033981 >= 0.0001
Iteration 132 (max 500)
   eps = 0.033812 >= 0.0001
Iteration 133 (max 500)
```

```
   eps = 0.033613 >= 0.0001
Iteration 134 (max 500)
   eps = 0.033338 >= 0.0001
Iteration 135 (max 500)
   eps = 0.033147 >= 0.0001
Iteration 136 (max 500)
   eps = 0.032955 >= 0.0001
Iteration 137 (max 500)
   eps = 0.032827 >= 0.0001
Iteration 138 (max 500)
   eps = 0.032688 >= 0.0001
Iteration 139 (max 500)
   eps = 0.032416 >= 0.0001
Iteration 140 (max 500)
   eps = 0.032171 >= 0.0001
Iteration 141 (max 500)
   eps = 0.031972 >= 0.0001
Iteration 142 (max 500)
   eps = 0.031739 >= 0.0001
Iteration 143 (max 500)
   eps = 0.03153 >= 0.0001
Iteration 144 (max 500)
   eps = 0.031352 >= 0.0001
Iteration 145 (max 500)
   eps = 0.031138 >= 0.0001
Iteration 146 (max 500)
   eps = 0.030897 >= 0.0001
Iteration 147 (max 500)
   eps = 0.030736 >= 0.0001
Iteration 148 (max 500)
   eps = 0.030574 >= 0.0001
Iteration 149 (max 500)
   eps = 0.030403 >= 0.0001
Iteration 150 (max 500)
   eps = 0.030253 >= 0.0001
Iteration 151 (max 500)
   eps = 0.0301 >= 0.0001
Iteration 152 (max 500)
   eps = 0.02997 >= 0.0001
Iteration 153 (max 500)
   eps = 0.029906 >= 0.0001
Iteration 154 (max 500)
   eps = 0.02981 >= 0.0001
Iteration 155 (max 500)
   eps = 0.029729 >= 0.0001
Iteration 156 (max 500)
   eps = 0.029618 >= 0.0001
Iteration 157 (max 500)
```

```
   eps = 0.029496 >= 0.0001
Iteration 158 (max 500)
   eps = 0.029387 >= 0.0001
Iteration 159 (max 500)
   eps = 0.029307 >= 0.0001
Iteration 160 (max 500)
   eps = 0.029155 >= 0.0001
Iteration 161 (max 500)
   eps = 0.029074 >= 0.0001
Iteration 162 (max 500)
   eps = 0.028941 >= 0.0001
Iteration 163 (max 500)
   eps = 0.028861 >= 0.0001
Iteration 164 (max 500)
   eps = 0.028722 >= 0.0001
Iteration 165 (max 500)
   eps = 0.028608 >= 0.0001
Iteration 166 (max 500)
   eps = 0.028474 >= 0.0001
Iteration 167 (max 500)
   eps = 0.028392 >= 0.0001
Iteration 168 (max 500)
   eps = 0.0283 >= 0.0001
Iteration 169 (max 500)
   eps = 0.02817 >= 0.0001
Iteration 170 (max 500)
   eps = 0.028087 >= 0.0001
Iteration 171 (max 500)
   eps = 0.028001 >= 0.0001
Iteration 172 (max 500)
   eps = 0.027931 >= 0.0001
Iteration 173 (max 500)
   eps = 0.027791 >= 0.0001
Iteration 174 (max 500)
   eps = 0.027657 >= 0.0001
Iteration 175 (max 500)
   eps = 0.027508 >= 0.0001
Iteration 176 (max 500)
   eps = 0.027448 >= 0.0001
Iteration 177 (max 500)
   eps = 0.027296 >= 0.0001
Iteration 178 (max 500)
   eps = 0.027217 >= 0.0001
Iteration 179 (max 500)
   eps = 0.027133 >= 0.0001
Iteration 180 (max 500)
   eps = 0.027024 >= 0.0001
Iteration 181 (max 500)
```

```
   eps = 0.026893 >= 0.0001
Iteration 182 (max 500)
   eps = 0.026779 >= 0.0001
Iteration 183 (max 500)
   eps = 0.026693 >= 0.0001
Iteration 184 (max 500)
   eps = 0.026576 >= 0.0001
Iteration 185 (max 500)
   eps = 0.026506 >= 0.0001
Iteration 186 (max 500)
   eps = 0.026393 >= 0.0001
Iteration 187 (max 500)
   eps = 0.026268 >= 0.0001
Iteration 188 (max 500)
   eps = 0.026166 >= 0.0001
Iteration 189 (max 500)
   eps = 0.026073 >= 0.0001
Iteration 190 (max 500)
   eps = 0.025934 >= 0.0001
Iteration 191 (max 500)
   eps = 0.025835 >= 0.0001
Iteration 192 (max 500)
   eps = 0.025708 >= 0.0001
Iteration 193 (max 500)
   eps = 0.025573 >= 0.0001
Iteration 194 (max 500)
   eps = 0.025476 >= 0.0001
Iteration 195 (max 500)
   eps = 0.025371 >= 0.0001
Iteration 196 (max 500)
   eps = 0.02527 >= 0.0001
Iteration 197 (max 500)
   eps = 0.025183 >= 0.0001
Iteration 198 (max 500)
   eps = 0.02515 >= 0.0001
Iteration 199 (max 500)
   eps = 0.025057 >= 0.0001
Iteration 200 (max 500)
   eps = 0.024937 >= 0.0001
Iteration 201 (max 500)
   eps = 0.024798 >= 0.0001
Iteration 202 (max 500)
   eps = 0.024746 >= 0.0001
Iteration 203 (max 500)
   eps = 0.024557 >= 0.0001
Iteration 204 (max 500)
   eps = 0.024437 >= 0.0001
Iteration 205 (max 500)
```

```
   eps = 0.024384 >= 0.0001
Iteration 206 (max 500)
   eps = 0.024265 >= 0.0001
Iteration 207 (max 500)
   eps = 0.024225 >= 0.0001
Iteration 208 (max 500)
   eps = 0.024219 >= 0.0001
Iteration 209 (max 500)
   eps = 0.0242 >= 0.0001
Iteration 210 (max 500)
   eps = 0.024113 >= 0.0001
Iteration 211 (max 500)
   eps = 0.023948 >= 0.0001
Iteration 212 (max 500)
   eps = 0.023769 >= 0.0001
Iteration 213 (max 500)
   eps = 0.023631 >= 0.0001
Iteration 214 (max 500)
   eps = 0.023472 >= 0.0001
Iteration 215 (max 500)
   eps = 0.023325 >= 0.0001
Iteration 216 (max 500)
   eps = 0.023201 >= 0.0001
Iteration 217 (max 500)
   eps = 0.02303 >= 0.0001
Iteration 218 (max 500)
   eps = 0.022848 >= 0.0001
Iteration 219 (max 500)
   eps = 0.022704 >= 0.0001
Iteration 220 (max 500)
   eps = 0.022577 >= 0.0001
Iteration 221 (max 500)
   eps = 0.022446 >= 0.0001
Iteration 222 (max 500)
   eps = 0.022349 >= 0.0001
Iteration 223 (max 500)
   eps = 0.022282 >= 0.0001
Iteration 224 (max 500)
   eps = 0.022167 >= 0.0001
Iteration 225 (max 500)
   eps = 0.022055 >= 0.0001
Iteration 226 (max 500)
   eps = 0.021952 >= 0.0001
Iteration 227 (max 500)
   eps = 0.021838 >= 0.0001
Iteration 228 (max 500)
   eps = 0.021763 >= 0.0001
Iteration 229 (max 500)
```

```
   eps = 0.021671 >= 0.0001
Iteration 230 (max 500)
   eps = 0.021576 >= 0.0001
Iteration 231 (max 500)
   eps = 0.02141 >= 0.0001
Iteration 232 (max 500)
   eps = 0.021271 >= 0.0001
Iteration 233 (max 500)
   eps = 0.021161 >= 0.0001
Iteration 234 (max 500)
   eps = 0.021056 >= 0.0001
Iteration 235 (max 500)
   eps = 0.020982 >= 0.0001
Iteration 236 (max 500)
   eps = 0.020866 >= 0.0001
Iteration 237 (max 500)
   eps = 0.02072 >= 0.0001
Iteration 238 (max 500)
   eps = 0.020601 >= 0.0001
Iteration 239 (max 500)
   eps = 0.020499 >= 0.0001
Iteration 240 (max 500)
   eps = 0.020382 >= 0.0001
Iteration 241 (max 500)
   eps = 0.02028 >= 0.0001
Iteration 242 (max 500)
   eps = 0.020205 >= 0.0001
Iteration 243 (max 500)
   eps = 0.020097 >= 0.0001
Iteration 244 (max 500)
   eps = 0.019988 >= 0.0001
Iteration 245 (max 500)
   eps = 0.019862 >= 0.0001
Iteration 246 (max 500)
   eps = 0.019746 >= 0.0001
Iteration 247 (max 500)
   eps = 0.019665 >= 0.0001
Iteration 248 (max 500)
   eps = 0.019589 >= 0.0001
Iteration 249 (max 500)
   eps = 0.019482 >= 0.0001
Iteration 250 (max 500)
   eps = 0.019402 >= 0.0001
Iteration 251 (max 500)
   eps = 0.019318 >= 0.0001
Iteration 252 (max 500)
   eps = 0.019206 >= 0.0001
Iteration 253 (max 500)
```

```
   eps = 0.019099 >= 0.0001
Iteration 254 (max 500)
   eps = 0.019022 >= 0.0001
Iteration 255 (max 500)
   eps = 0.01894 >= 0.0001
Iteration 256 (max 500)
   eps = 0.018853 >= 0.0001
Iteration 257 (max 500)
   eps = 0.018735 >= 0.0001
Iteration 258 (max 500)
   eps = 0.018654 >= 0.0001
Iteration 259 (max 500)
   eps = 0.018572 >= 0.0001
Iteration 260 (max 500)
   eps = 0.018496 >= 0.0001
Iteration 261 (max 500)
   eps = 0.01842 >= 0.0001
Iteration 262 (max 500)
   eps = 0.018358 >= 0.0001
Iteration 263 (max 500)
   eps = 0.018274 >= 0.0001
Iteration 264 (max 500)
   eps = 0.018214 >= 0.0001
Iteration 265 (max 500)
   eps = 0.018131 >= 0.0001
Iteration 266 (max 500)
   eps = 0.01806 >= 0.0001
Iteration 267 (max 500)
   eps = 0.017982 >= 0.0001
Iteration 268 (max 500)
   eps = 0.017911 >= 0.0001
Iteration 269 (max 500)
   eps = 0.017838 >= 0.0001
Iteration 270 (max 500)
   eps = 0.017775 >= 0.0001
Iteration 271 (max 500)
   eps = 0.017664 >= 0.0001
Iteration 272 (max 500)
   eps = 0.017554 >= 0.0001
Iteration 273 (max 500)
   eps = 0.017479 >= 0.0001
Iteration 274 (max 500)
   eps = 0.017398 >= 0.0001
Iteration 275 (max 500)
   eps = 0.017324 >= 0.0001
Iteration 276 (max 500)
   eps = 0.017223 >= 0.0001
Iteration 277 (max 500)
```

```
   eps = 0.017124 >= 0.0001
Iteration 278 (max 500)
   eps = 0.017041 >= 0.0001
Iteration 279 (max 500)
   eps = 0.01696 >= 0.0001
Iteration 280 (max 500)
   eps = 0.016892 >= 0.0001
Iteration 281 (max 500)
   eps = 0.016818 >= 0.0001
Iteration 282 (max 500)
   eps = 0.016761 >= 0.0001
Iteration 283 (max 500)
   eps = 0.016677 >= 0.0001
Iteration 284 (max 500)
   eps = 0.016593 >= 0.0001
Iteration 285 (max 500)
   eps = 0.016486 >= 0.0001
Iteration 286 (max 500)
   eps = 0.016417 >= 0.0001
Iteration 287 (max 500)
   eps = 0.016322 >= 0.0001
Iteration 288 (max 500)
   eps = 0.016277 >= 0.0001
Iteration 289 (max 500)
   eps = 0.016224 >= 0.0001
Iteration 290 (max 500)
   eps = 0.016135 >= 0.0001
Iteration 291 (max 500)
   eps = 0.016045 >= 0.0001
Iteration 292 (max 500)
   eps = 0.015979 >= 0.0001
Iteration 293 (max 500)
   eps = 0.015922 >= 0.0001
Iteration 294 (max 500)
   eps = 0.015834 >= 0.0001
Iteration 295 (max 500)
   eps = 0.015763 >= 0.0001
Iteration 296 (max 500)
   eps = 0.015697 >= 0.0001
Iteration 297 (max 500)
   eps = 0.015618 >= 0.0001
Iteration 298 (max 500)
   eps = 0.015573 >= 0.0001
Iteration 299 (max 500)
   eps = 0.015494 >= 0.0001
Iteration 300 (max 500)
   eps = 0.015392 >= 0.0001
Iteration 301 (max 500)
```

```
   eps = 0.015316 >= 0.0001
Iteration 302 (max 500)
   eps = 0.01525 >= 0.0001
Iteration 303 (max 500)
   eps = 0.015182 >= 0.0001
Iteration 304 (max 500)
   eps = 0.015135 >= 0.0001
Iteration 305 (max 500)
   eps = 0.015062 >= 0.0001
Iteration 306 (max 500)
   eps = 0.014996 >= 0.0001
Iteration 307 (max 500)
   eps = 0.014913 >= 0.0001
Iteration 308 (max 500)
   eps = 0.014855 >= 0.0001
Iteration 309 (max 500)
   eps = 0.014781 >= 0.0001
Iteration 310 (max 500)
   eps = 0.014706 >= 0.0001
Iteration 311 (max 500)
   eps = 0.014654 >= 0.0001
Iteration 312 (max 500)
   eps = 0.01458 >= 0.0001
Iteration 313 (max 500)
   eps = 0.01451 >= 0.0001
Iteration 314 (max 500)
   eps = 0.014447 >= 0.0001
Iteration 315 (max 500)
   eps = 0.014369 >= 0.0001
Iteration 316 (max 500)
   eps = 0.014293 >= 0.0001
Iteration 317 (max 500)
   eps = 0.014233 >= 0.0001
Iteration 318 (max 500)
   eps = 0.014169 >= 0.0001
Iteration 319 (max 500)
   eps = 0.014119 >= 0.0001
Iteration 320 (max 500)
   eps = 0.014078 >= 0.0001
Iteration 321 (max 500)
   eps = 0.014031 >= 0.0001
Iteration 322 (max 500)
   eps = 0.013972 >= 0.0001
Iteration 323 (max 500)
   eps = 0.0139 >= 0.0001
Iteration 324 (max 500)
   eps = 0.013835 >= 0.0001
Iteration 325 (max 500)
```

```
   eps = 0.01375 >= 0.0001
Iteration 326 (max 500)
   eps = 0.013692 >= 0.0001
Iteration 327 (max 500)
   eps = 0.013611 >= 0.0001
Iteration 328 (max 500)
   eps = 0.013539 >= 0.0001
Iteration 329 (max 500)
   eps = 0.013454 >= 0.0001
Iteration 330 (max 500)
   eps = 0.013392 >= 0.0001
Iteration 331 (max 500)
   eps = 0.013322 >= 0.0001
Iteration 332 (max 500)
   eps = 0.013245 >= 0.0001
Iteration 333 (max 500)
   eps = 0.013175 >= 0.0001
Iteration 334 (max 500)
   eps = 0.013107 >= 0.0001
Iteration 335 (max 500)
   eps = 0.01305 >= 0.0001
Iteration 336 (max 500)
   eps = 0.012994 >= 0.0001
Iteration 337 (max 500)
   eps = 0.01294 >= 0.0001
Iteration 338 (max 500)
   eps = 0.01288 >= 0.0001
Iteration 339 (max 500)
   eps = 0.012824 >= 0.0001
Iteration 340 (max 500)
   eps = 0.01277 >= 0.0001
Iteration 341 (max 500)
   eps = 0.012711 >= 0.0001
Iteration 342 (max 500)
   eps = 0.01266 >= 0.0001
Iteration 343 (max 500)
   eps = 0.012607 >= 0.0001
Iteration 344 (max 500)
   eps = 0.012538 >= 0.0001
Iteration 345 (max 500)
   eps = 0.012468 >= 0.0001
Iteration 346 (max 500)
   eps = 0.012401 >= 0.0001
Iteration 347 (max 500)
   eps = 0.012338 >= 0.0001
Iteration 348 (max 500)
   eps = 0.01226 >= 0.0001
Iteration 349 (max 500)
```

```
   eps = 0.012169 >= 0.0001
Iteration 350 (max 500)
   eps = 0.012096 >= 0.0001
Iteration 351 (max 500)
   eps = 0.012018 >= 0.0001
Iteration 352 (max 500)
   eps = 0.011956 >= 0.0001
Iteration 353 (max 500)
   eps = 0.0119 >= 0.0001
Iteration 354 (max 500)
   eps = 0.011843 >= 0.0001
Iteration 355 (max 500)
   eps = 0.011797 >= 0.0001
Iteration 356 (max 500)
   eps = 0.011749 >= 0.0001
Iteration 357 (max 500)
   eps = 0.011701 >= 0.0001
Iteration 358 (max 500)
   eps = 0.011649 >= 0.0001
Iteration 359 (max 500)
   eps = 0.011591 >= 0.0001
Iteration 360 (max 500)
   eps = 0.011533 >= 0.0001
Iteration 361 (max 500)
   eps = 0.011475 >= 0.0001
Iteration 362 (max 500)
   eps = 0.011423 >= 0.0001
Iteration 363 (max 500)
   eps = 0.011368 >= 0.0001
Iteration 364 (max 500)
   eps = 0.011314 >= 0.0001
Iteration 365 (max 500)
   eps = 0.01126 >= 0.0001
Iteration 366 (max 500)
   eps = 0.011217 >= 0.0001
Iteration 367 (max 500)
   eps = 0.011163 >= 0.0001
Iteration 368 (max 500)
   eps = 0.011111 >= 0.0001
Iteration 369 (max 500)
   eps = 0.011057 >= 0.0001
Iteration 370 (max 500)
   eps = 0.011005 >= 0.0001
Iteration 371 (max 500)
   eps = 0.010953 >= 0.0001
Iteration 372 (max 500)
   eps = 0.010904 >= 0.0001
Iteration 373 (max 500)
```

```
   eps = 0.010848 >= 0.0001
Iteration 374 (max 500)
   eps = 0.010788 >= 0.0001
Iteration 375 (max 500)
   eps = 0.010735 >= 0.0001
Iteration 376 (max 500)
   eps = 0.01068 >= 0.0001
Iteration 377 (max 500)
   eps = 0.010626 >= 0.0001
Iteration 378 (max 500)
   eps = 0.010574 >= 0.0001
Iteration 379 (max 500)
   eps = 0.010526 >= 0.0001
Iteration 380 (max 500)
   eps = 0.010476 >= 0.0001
Iteration 381 (max 500)
   eps = 0.010427 >= 0.0001
Iteration 382 (max 500)
   eps = 0.010378 >= 0.0001
Iteration 383 (max 500)
   eps = 0.010336 >= 0.0001
Iteration 384 (max 500)
   eps = 0.010295 >= 0.0001
Iteration 385 (max 500)
   eps = 0.010255 >= 0.0001
Iteration 386 (max 500)
   eps = 0.010214 >= 0.0001
Iteration 387 (max 500)
   eps = 0.010161 >= 0.0001
Iteration 388 (max 500)
   eps = 0.010118 >= 0.0001
Iteration 389 (max 500)
   eps = 0.010072 >= 0.0001
Iteration 390 (max 500)
   eps = 0.010017 >= 0.0001
Iteration 391 (max 500)
   eps = 0.009955 >= 0.0001
Iteration 392 (max 500)
   eps = 0.009898 >= 0.0001
Iteration 393 (max 500)
   eps = 0.009844 >= 0.0001
Iteration 394 (max 500)
   eps = 0.009801 >= 0.0001
Iteration 395 (max 500)
   eps = 0.009751 >= 0.0001
Iteration 396 (max 500)
   eps = 0.009694 >= 0.0001
Iteration 397 (max 500)
```

```
   eps = 0.009647 >= 0.0001
Iteration 398 (max 500)
   eps = 0.009604 >= 0.0001
Iteration 399 (max 500)
   eps = 0.009561 >= 0.0001
Iteration 400 (max 500)
   eps = 0.00951 >= 0.0001
Iteration 401 (max 500)
   eps = 0.009476 >= 0.0001
Iteration 402 (max 500)
   eps = 0.00943 >= 0.0001
Iteration 403 (max 500)
   eps = 0.009383 >= 0.0001
Iteration 404 (max 500)
   eps = 0.009331 >= 0.0001
Iteration 405 (max 500)
   eps = 0.009285 >= 0.0001
Iteration 406 (max 500)
   eps = 0.009239 >= 0.0001
Iteration 407 (max 500)
   eps = 0.0092 >= 0.0001
Iteration 408 (max 500)
   eps = 0.00916 >= 0.0001
Iteration 409 (max 500)
   eps = 0.009134 >= 0.0001
Iteration 410 (max 500)
   eps = 0.00912 >= 0.0001
Iteration 411 (max 500)
   eps = 0.009091 >= 0.0001
Iteration 412 (max 500)
   eps = 0.009059 >= 0.0001
Iteration 413 (max 500)
   eps = 0.008997 >= 0.0001
Iteration 414 (max 500)
   eps = 0.008947 >= 0.0001
Iteration 415 (max 500)
   eps = 0.008896 >= 0.0001
Iteration 416 (max 500)
   eps = 0.008841 >= 0.0001
Iteration 417 (max 500)
   eps = 0.008786 >= 0.0001
Iteration 418 (max 500)
   eps = 0.008739 >= 0.0001
Iteration 419 (max 500)
   eps = 0.008697 >= 0.0001
Iteration 420 (max 500)
   eps = 0.008656 >= 0.0001
Iteration 421 (max 500)
```

```
   eps = 0.008623 >= 0.0001
Iteration 422 (max 500)
   eps = 0.008586 >= 0.0001
Iteration 423 (max 500)
   eps = 0.008547 >= 0.0001
Iteration 424 (max 500)
   eps = 0.008499 >= 0.0001
Iteration 425 (max 500)
   eps = 0.008452 >= 0.0001
Iteration 426 (max 500)
   eps = 0.008402 >= 0.0001
Iteration 427 (max 500)
   eps = 0.008353 >= 0.0001
Iteration 428 (max 500)
   eps = 0.008317 >= 0.0001
Iteration 429 (max 500)
   eps = 0.008275 >= 0.0001
Iteration 430 (max 500)
   eps = 0.008231 >= 0.0001
Iteration 431 (max 500)
   eps = 0.008188 >= 0.0001
Iteration 432 (max 500)
   eps = 0.008157 >= 0.0001
Iteration 433 (max 500)
   eps = 0.008124 >= 0.0001
Iteration 434 (max 500)
   eps = 0.008084 >= 0.0001
Iteration 435 (max 500)
   eps = 0.008053 >= 0.0001
Iteration 436 (max 500)
   eps = 0.008027 >= 0.0001
Iteration 437 (max 500)
   eps = 0.007994 >= 0.0001
Iteration 438 (max 500)
   eps = 0.007961 >= 0.0001
Iteration 439 (max 500)
   eps = 0.007924 >= 0.0001
Iteration 440 (max 500)
   eps = 0.0079 >= 0.0001
Iteration 441 (max 500)
   eps = 0.007858 >= 0.0001
Iteration 442 (max 500)
   eps = 0.007827 >= 0.0001
Iteration 443 (max 500)
   eps = 0.007786 >= 0.0001
Iteration 444 (max 500)
   eps = 0.007784 >= 0.0001
Iteration 445 (max 500)
```

```
   eps = 0.007738 >= 0.0001
Iteration 446 (max 500)
   eps = 0.007702 >= 0.0001
Iteration 447 (max 500)
   eps = 0.007659 >= 0.0001
Iteration 448 (max 500)
   eps = 0.007625 >= 0.0001
Iteration 449 (max 500)
   eps = 0.007592 >= 0.0001
Iteration 450 (max 500)
   eps = 0.007624 >= 0.0001
Iteration 451 (max 500)
   eps = 0.007613 >= 0.0001
Iteration 452 (max 500)
   eps = 0.007533 >= 0.0001
Iteration 453 (max 500)
   eps = 0.007489 >= 0.0001
Iteration 454 (max 500)
   eps = 0.007438 >= 0.0001
Iteration 455 (max 500)
   eps = 0.007393 >= 0.0001
Iteration 456 (max 500)
   eps = 0.007351 >= 0.0001
Iteration 457 (max 500)
   eps = 0.007317 >= 0.0001
Iteration 458 (max 500)
   eps = 0.007271 >= 0.0001
Iteration 459 (max 500)
   eps = 0.00729 >= 0.0001
Iteration 460 (max 500)
   eps = 0.00724 >= 0.0001
Iteration 461 (max 500)
   eps = 0.007202 >= 0.0001
Iteration 462 (max 500)
   eps = 0.007161 >= 0.0001
Iteration 463 (max 500)
   eps = 0.007146 >= 0.0001
Iteration 464 (max 500)
   eps = 0.007109 >= 0.0001
Iteration 465 (max 500)
   eps = 0.00708 >= 0.0001
Iteration 466 (max 500)
   eps = 0.007049 >= 0.0001
Iteration 467 (max 500)
   eps = 0.007015 >= 0.0001
Iteration 468 (max 500)
   eps = 0.006982 >= 0.0001
Iteration 469 (max 500)
```

```
   eps = 0.00695 >= 0.0001
Iteration 470 (max 500)
   eps = 0.00691 >= 0.0001
Iteration 471 (max 500)
   eps = 0.006874 >= 0.0001
Iteration 472 (max 500)
   eps = 0.006846 >= 0.0001
Iteration 473 (max 500)
   eps = 0.006807 >= 0.0001
Iteration 474 (max 500)
   eps = 0.006775 >= 0.0001
Iteration 475 (max 500)
   eps = 0.006737 >= 0.0001
Iteration 476 (max 500)
   eps = 0.006698 >= 0.0001
Iteration 477 (max 500)
   eps = 0.006674 >= 0.0001
Iteration 478 (max 500)
   eps = 0.006639 >= 0.0001
Iteration 479 (max 500)
   eps = 0.00661 >= 0.0001
Iteration 480 (max 500)
   eps = 0.006572 >= 0.0001
Iteration 481 (max 500)
   eps = 0.006541 >= 0.0001
Iteration 482 (max 500)
   eps = 0.00651 >= 0.0001
Iteration 483 (max 500)
   eps = 0.006483 >= 0.0001
Iteration 484 (max 500)
   eps = 0.006445 >= 0.0001
Iteration 485 (max 500)
   eps = 0.006407 >= 0.0001
Iteration 486 (max 500)
   eps = 0.006451 >= 0.0001
Iteration 487 (max 500)
   eps = 0.006369 >= 0.0001
Iteration 488 (max 500)
   eps = 0.006327 >= 0.0001
Iteration 489 (max 500)
   eps = 0.006291 >= 0.0001
Iteration 490 (max 500)
   eps = 0.006255 >= 0.0001
Iteration 491 (max 500)
   eps = 0.006222 >= 0.0001
Iteration 492 (max 500)
   eps = 0.006199 >= 0.0001
Iteration 493 (max 500)
```

```
   eps = 0.006163 >= 0.0001
Iteration 494 (max 500)
   eps = 0.006137 >= 0.0001
Iteration 495 (max 500)
   eps = 0.006106 >= 0.0001
Iteration 496 (max 500)
   eps = 0.006071 >= 0.0001
Iteration 497 (max 500)
   eps = 0.00604 >= 0.0001
Iteration 498 (max 500)
   eps = 0.00601 >= 0.0001
Iteration 499 (max 500)
   eps = 0.005984 >= 0.0001
Iteration 500 (max 500)
   eps = 0.005953 >= 0.0001
Time spent by the wind speed solver: 53.4724280834198 s
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Rotates geometries from -50.0 degrees
Processing UMEP EXIT umep:Urban Wind Field: URock
```

### 4.7 Post-traitement : Tif to netCDF

[42]:

```
import numpy as np
import rasterio
from pyproj import Transformer

import os

TMRT_PATH = os.path.join(output_path, 'Tmrt_average.tif')
with rasterio.open(TMRT_PATH) as tif:
    temp = tif.read(1)
    tif_transform = tif.transform
    tif_crs = tif.crs
    tif_width = tif.width
    tif_height = tif.height
    print('tif crs, tif transform', tif_crs, tif_transform)
    print('tif width, tif height:', tif_width, tif_height)
```

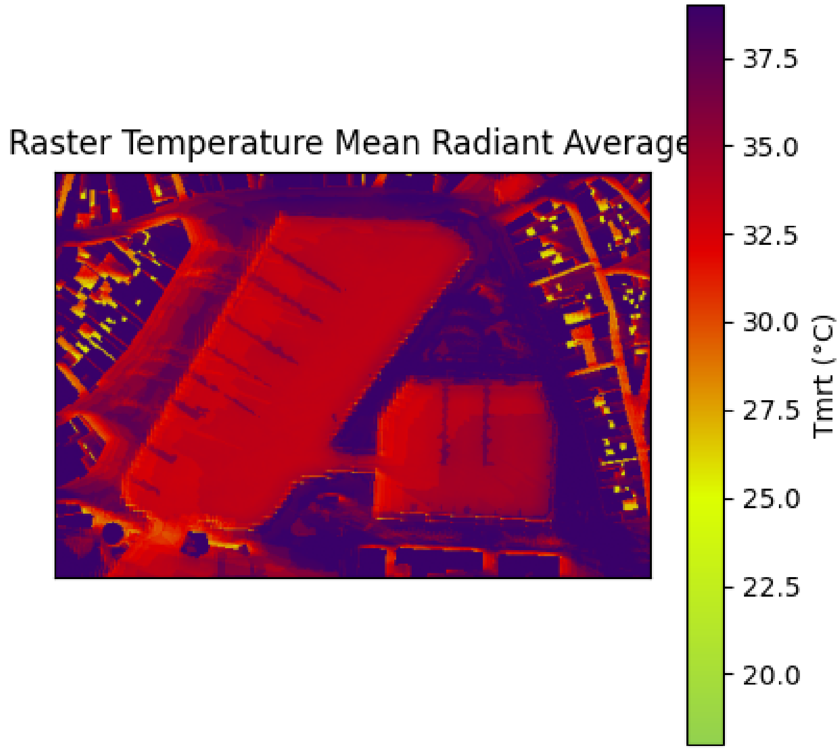
```
    rows, cols = np.meshgrid(np.arange(tif_height), np.arange(tif_width),␣
↪indexing='ij')
    xs, ys = rasterio.transform.xy(tif_transform, rows, cols)
    xs = np.array(xs)
    ys = np.array(ys)

# Flatten for reprojection
xs_flat = xs.flatten()
ys_flat = ys.flatten()

# Set up a transformer from EPSG:2154 to WGS84
transformer = Transformer.from_crs(tif_crs, "EPSG:4326", always_xy=True)
lons_flat, lats_flat = transformer.transform(xs_flat, ys_flat)

# Reshape to tif grid shape
lons = lons_flat.reshape(xs.shape)
lats = lats_flat.reshape(ys.shape)
```

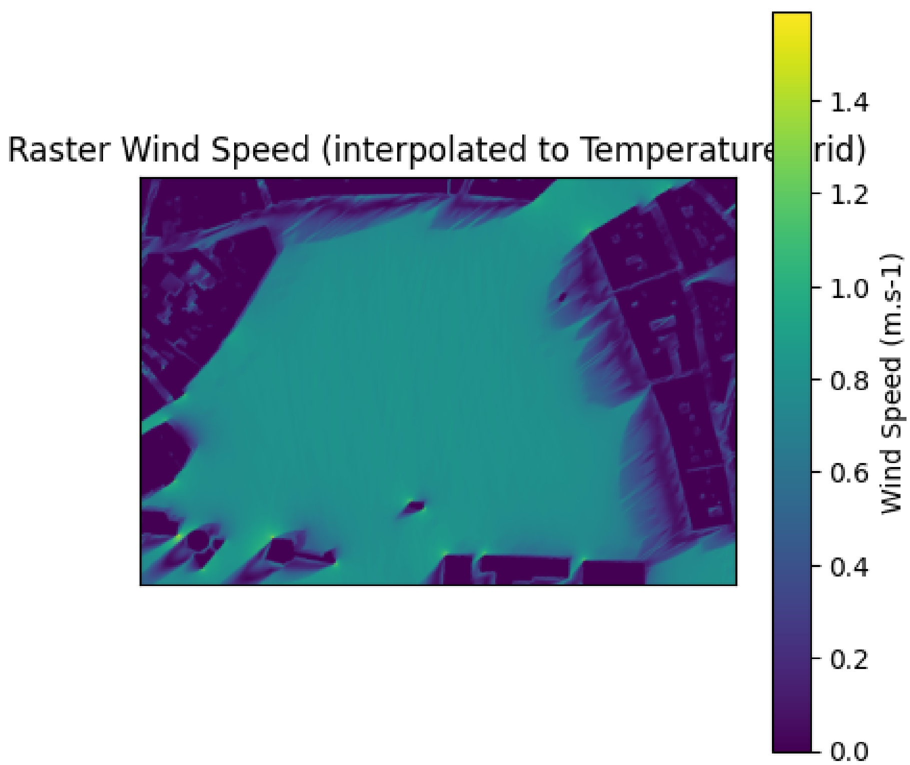
```
tif crs, tif transform EPSG:2154 | 1.00, 0.00, 379509.08|
| 0.00,-1.00, 6570483.59|
| 0.00, 0.00, 1.00|
tif width, tif height: 453 309
```

[43]:
```
from netCDF4 import Dataset
import numpy as np

direction = 50
file_name = f'output_{direction}.nc'
file_nc = os.path.join(output_path_urock, file_name)

nc = Dataset(file_nc)
group = nc.groups['3D_wind']

lon = group.variables['lon'][:]
lat = group.variables['lat'][:]

Z = group.variables['Z'][:]
level_idx = np.argmin(np.abs(Z - 10))  # 10 meter

wind_x = group.variables['windSpeed_x'][:, :, level_idx]
wind_y = group.variables['windSpeed_y'][:, :, level_idx]
wind_z = group.variables['windSpeed_z'][:, :, level_idx]

wind_speed = np.sqrt(wind_x ** 2 + wind_y ** 2 + wind_z ** 2)

# flatten for interpolation
```

```python
points = np.column_stack((lon.flatten(), lat.flatten()))
values = wind_speed.flatten()

from scipy.interpolate import griddata

# interpolate from (lons,lats) from TIF
wind_speed_on_tif = griddata(points, values, (lons, lats), method='linear')
wind_as_array = wind_speed_on_tif.reshape(tif_height, tif_width)
```

[44]:
```python
from matplotlib.colors import LinearSegmentedColormap, Normalize
%matplotlib inline

boundaries = [18, 21, 28, 35, 39]
colors = ['#92d14f', '#ddff00', '#e10000', '#390069']
cmap = LinearSegmentedColormap.from_list('custom_cmap', colors, N=256)
norm = Normalize(vmin=min(boundaries), vmax=max(boundaries))

fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
img = plt.imshow(temp, cmap=cmap, norm=norm)
plt.colorbar(img, ax=ax, label='Tmrt (°C)')
plt.title('Raster Temperature Mean Radiant Average')
plt.show()

fig, ax = plt.subplots(figsize=(5, 5))
ax.set_xticks([])
ax.set_yticks([])
img = plt.imshow(wind_as_array)
plt.colorbar(img, ax=ax, label='Wind Speed (m.s-1)')
plt.title('Raster Wind Speed (interpolated to Temperature grid)')
plt.show()
```


Raster Temperature Mean Radiant Average
37.5
35.0
32.5
30.0
27.5
25.0
22.5
20.0
Tmrt (°C)


Raster Wind Speed (interpolated to Temperature Grid)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Wind Speed (m.s-1)

### 4.8 Calcul de l'UTCI : (Universal Thermal Climate Index)

[45]:
```
from pymdu.physics.umep.UmepCore import UmepCore

src_nc = os.path.join(output_path_urock, f'output_{direction}.nc')
umep_core = UmepCore(output_dir=output_path_urock)

options_umep_urock_analyze = {
    'INPUT_LINES': None,
    'IS_STREAM': False,
    'ID_FIELD_LINES': '',
    'INPUT_POLYGONS': None,
    'ID_FIELD_POLYGONS': '',
    'INPUT_WIND_FILE': src_nc,
    'SIMULATION_NAME': '',
    'OUTPUT_DIRECTORY': output_path_urock
}
umep_core.run_processing(
    name="umep:Urban Wind Field: URock analyzer",
    options=options_umep_urock_analyze
)
```

```
__init__ QGisCore
__init__ qgsApp
platform.system() Darwin
__init__ UmepCore
Processing UMEP umep:Urban Wind Field: URock analyzer
{'INPUT_LINES': None, 'IS_STREAM': False, 'ID_FIELD_LINES': '',
'INPUT_POLYGONS': None, 'ID_FIELD_POLYGONS': '', 'INPUT_WIND_FILE': '/Users/Bori
s/Documents/TIPEE/pymdu/demos/results_demo/output_urock/output_50.nc',
'SIMULATION_NAME': '', 'OUTPUT_DIRECTORY':
'/Users/Boris/Documents/TIPEE/pymdu/demos/results_demo/output_urock'}
Connecting to database
->/var/folders/zh/f2j36cz90lzfcn8r42snc4nc0000gr/T/myDbH21750670969_068386
/Users/Boris/miniforge3/envs/pymdu/share/qgis/python/plugins/processing_umep/fun
ctions/URock/h2gis-standalone/h2gis-dist-2.2.3.jar
Connected!

Spatial functions added!

Processing UMEP EXIT umep:Urban Wind Field: URock analyzer
```

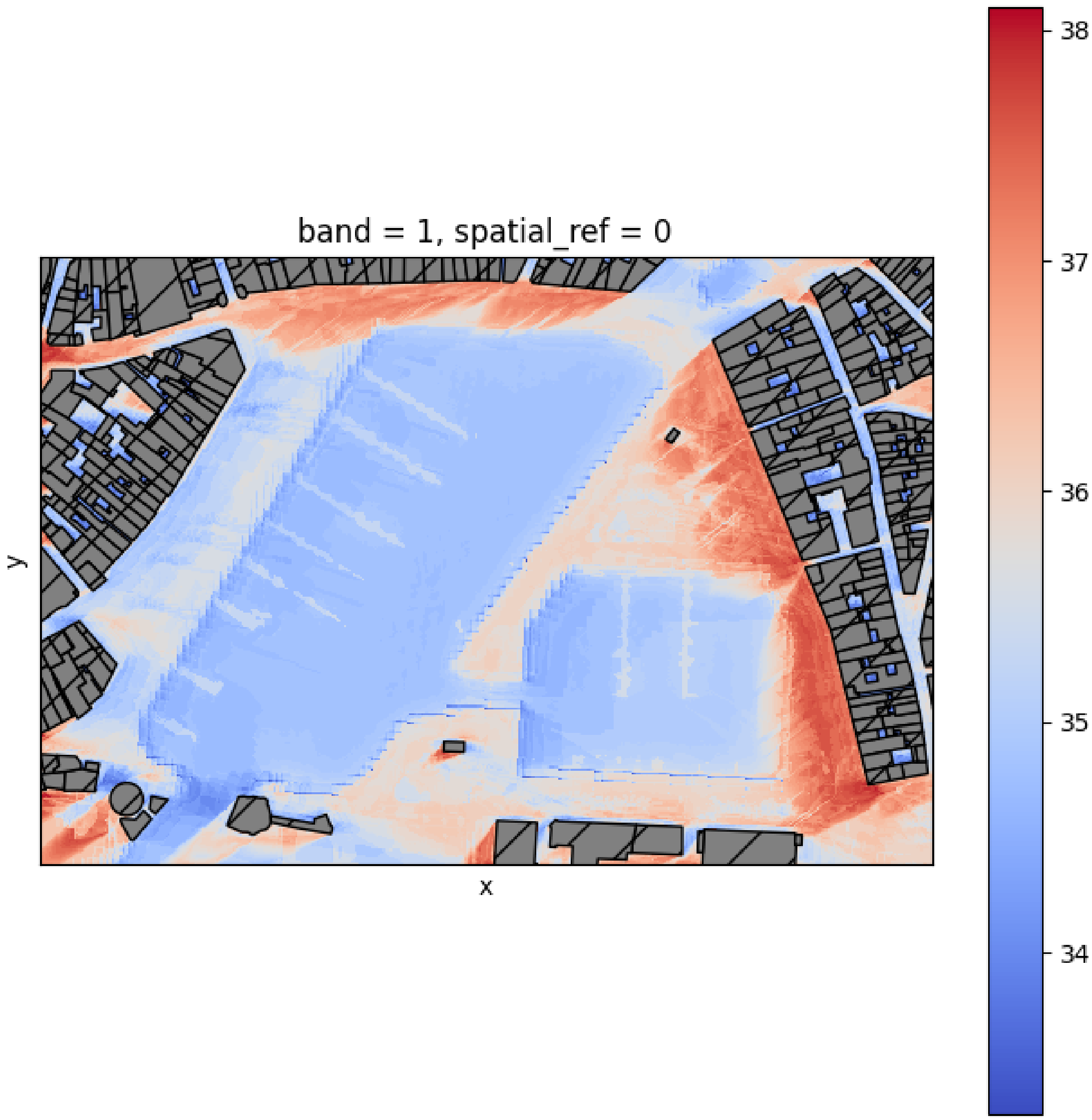
[46]:
```
import numpy as np
import pandas as pd
import rioxarray
```

```python
from tqdm import tqdm
from pythermalcomfort.models import utci
import os


def wind_a_10m_vectorized(x):
    return x * np.log(10 / 0.01) / np.log(np.minimum(1.5, 10) / 0.01)


output_path = os.path.join(os.getcwd(), 'results_demo')
output_path_urock = os.path.join(output_path, 'output_urock')
os.makedirs(output_path_urock, exist_ok=True)

METEO_FILE = 'FRA_AC_La.Rochelle.073150_TMYx.2004-2018.txt'
METEO_DATA = pd.read_csv(METEO_FILE, sep=' ')

direction = 50
wind_velocity = 4.1
HEURE = 16

TMRT_PATH = os.path.join(output_path, 'Tmrt_average.tif')
TMRT_dataset = rioxarray.open_rasterio(TMRT_PATH)
tmr_as_array = TMRT_dataset.data[0]
size1, size2 = tmr_as_array.shape

wind_as_array_speed = wind_velocity * wind_a_10m_vectorized(wind_as_array)

output = np.zeros(shape=(size1, size2))
tdb = METEO_DATA[METEO_DATA.it == HEURE].Td.values[0]
rh = METEO_DATA[METEO_DATA.it == HEURE].RH.values[0]
for i in tqdm(range(0, size1)):
    output[i, :] = utci(tdb=tdb, tr=tmr_as_array[i, :],␣
 ↪v=wind_as_array_speed[i, :], rh=rh, limit_inputs=False)
UTCI_dataset = TMRT_dataset.copy()
UTCI_dataset.data[0] = output
```

```
100%|
                                        | 309/309
[00:00<00:00, 10412.47it/s]
```

[50]:
```python
from matplotlib.colors import LinearSegmentedColormap, Normalize

inputs_simulation_path = os.path.join(os.getcwd(), 'results_demo/
 ↪inputs_simulation')

#boundaries = [21, 21, 28, 35, 42, 46, 53]
boundaries = [-40, -27, -13, 0, 9, 26, 32, 38, 46]
```

```
colors = ['#00007f', '#0301c1', '#0000fb', '#0061fe', '#01c0fd', '#00c000',␣
↪'#ff6601', '#ff3200', '#cc0001', '#7e0305']
cmap = LinearSegmentedColormap.from_list('custom_cmap', colors, N=10)
norm = Normalize(vmin=min(boundaries), vmax=max(boundaries))

fig, ax = plt.subplots(figsize=(8, 8))
ax.set_xticks([])
ax.set_yticks([])
plt.title("UTCI (Universal Thermal Climate Index)")
UTCI_dataset.plot(ax=ax, cmap="coolwarm", norm=None, add_colorbar=True)
# lidar_trees_gdf.to_crs(2154).plot(ax=ax, color='g', alpha=1)
buildings_gdf.plot(ax=ax, color='grey', edgecolor='k', hatch='/')
plt.show()
```



[ ]: