Using the `riskmapjnr` Python module to map the deforestation risk following the JNR methodology

First report for FAO focused on small-scale test areas
FAO budget code: TF/NFODD/TFGB110021450

Ghislain Vieilledent[*, ‡]

December 21, 2022

[*] **Correspondence to:** ghislain.vieilledent@cirad.fr
[‡] **CIRAD**, UMR AMAP, F-34398 Montpellier, FRANCE

The JNR (Jurisdictional Nested REDD+) provides a standardized approach to estimate and credit greenhouse gas (GHG) emission reductions in the framework of REDD+ (Reducing Emissions from Deforestation and Forest Degradation) programs. The JNR approach relies on a map of the deforestation risk to be able to allocate the deforestation spatially within the jurisdiction when projecting future deforestation under the reference scenario. The JNR Risk Mapping Tool presents a standardized approach to obtain this map of the deforestation risk. The methodology is based on minimal spatial information provided by the past deforestation map at the jurisdictional scale. While the methodology favors simplicity, the approach requires several computationally intensive geoprocessing steps on potentially large raster files. The `riskmapjnr` Python package we developed provides easy-to-use functions to obtain the risk map following the JNR methodology. Functions use optimized Python code that can efficiently process large raster files without memory issues. Here we provide an example of the use of the `riskmapjnr` Python package for small-case test areas. A first stable version (v1.0) of the package has been released at the end of the year 2022. Nonetheless, the package is fairly recent and **the code might contains some errors**. As a consequence, **these preliminary results should be interpreted with caution**.

Contents

List of Figures

1 Introduction

1.1 Context

In the forest sector, programs to mitigate GHG emissions across entire national or subnational jurisdictions (called REDD+ programs, i.e. programs aiming at Reducing Emissions from Deforestation and Forest Degradation) can be accounted for and credited using a jurisdictional-scale framework called JNR (Jurisdictional and Nested REDD+). JNR integrates government-led and project-level REDD+ activities and establishes a clear pathway for subnational- and project-level activities to be incorporated within broader REDD+ programs. The JNR framework ensures all projects and other reducing emissions from deforestation and degradation activities in a given jurisdiction are developed using consistent baselines and crediting approaches. They mitigate the risk of "leakage", i.e. the displacement of emission-causing activities to areas outside the project boundary, by monitoring emissions across an entire jurisdictional area. JNR is part of the VCS (Verified Carbon Standard) program which allows certified projects to turn their greenhouse gas (GHG) emission reductions and removals into tradable carbon credits. Since its launch in 2006, the VCS program has grown into the world's largest voluntary GHG program.

The JNR Risk Mapping Tool is a "benchmark" methodology that provides a standardized approach for developing deforestation and forest degradation risk maps for users of the JNR Allocation Tool in the context of JNR requirements. The methodology allows deriving a map of the deforestation (or degradation) risk based on a minimal spatial information provided by the past deforestation (or degradation) map at the jurisdictional scale. The JNR Risk Mapping Tool allows the creation of categorical and spatially static maps whose categories represent different levels of risk of deforestation or forest degradation in the validity period of the Forest Reference Emissions Level (FREL) and throughout the jurisdictional geographical boundaries. In the JNR Allocation Tool, the level of risk determines how the jurisdictional FREL is spatially distributed to nested lower-level jurisdictional programs and projects.

1.2 Objectives

While the JNR Risk Mapping Tool methodology favors simplicity, obtaining the risk map is not straightforward. The approach requires several geoprocessing steps on raster data that can be large, i.e. covering large spatial extent (eg. national scale) at high spatial resolution (eg. 30 m). The `riskmapjnr` Python package we developed includes functions to perform these geoprocessing steps and derive a risk map on any jurisdiction and at any spatial resolution following the JNR Risk Mapping Tool methodology. The `riskmapjnr` package includes functions to:

1. Estimate the distance to forest edge beyond which the deforestation risk is negligible: `dist_edge_threshold()`.

2. Compute local deforestation rates using a moving window whose size can vary: `local_defor_rate()`.
3. Transform local deforestation rates into categories of deforestation risks using several slicing algorithms: `set_defor_cat_zero()` and `defor_cat()`
4. Validate maps of deforestation risk and select the map with the higher accuracy: `defrate_per_cat()` and `validation()`.

The `riskmapjnr` package uses several known Python scientific packages such as `NumPy`, `SciPy`, and `Pandas` for fast matrix and vector operations and `gdal` for processing georeferenced raster data. Raster data are divided into blocks of data for in-memory processing. Such an approach allow processing large raster files with large geographical extents (e.g. country scale) and high spatial resolutions (eg. 30 m). Here we present an example of the use of the `riskmapjnr` Python package for small-case test areas following all the steps of the JNR methodology.

1.3 Warnings

The `riskmapjnr` package is still under development. The code might include some errors and still need to be thoroughly tested. A first stable version (v1.0) of the package will be released at the end of the year 2022. As a consequence, the preliminary results presented here should be interpreted with caution. Moreover, the JNR methodology is currently discussed and will be subject to changes in the coming months. Future versions of the `riskmapjnr` package should integrate these changes.

2 Initial setup and data

2.1 Installation

You need several dependencies to run the `riskmapjnr` Python package. The best way to install the package is to create a Python virtual environment, either through `conda` (recommended) or `virtualenv`.

1. Using `conda` (recommended)

   You first need to have `miniconda3` installed (see miniconda instructions). Then, create a conda environment (see conda environment instructions) and install the `riskmapjnr` package with the following commands:

```
conda create --name conda-rmj -c conda-forge python=3 gdal numpy
↪  matplotlib pandas pip scipy --yes
conda activate conda-rmj
pip install riskmapjnr # For PyPI version
```

```
# pip install
↪   https://github.com/ghislainv/riskmapjnr/archive/master.zip #
↪   For GitHub dev version
# conda install -c conda-forge jupyter geopandas descartes folium
↪   --yes  # Optional additional packages
```

To deactivate and delete the conda environment:

```
conda deactivate
conda env remove --name conda-rmj
```

2. Using `virtualenv`

   You first need to have the `virtualenv` package installed (see virtualenv instructions). Then, create a virtual environment and install the `riskmapjnr` package with the following commands:

```
cd ~
mkdir venvs # Directory for virtual environments
cd venvs
virtualenv --python=/usr/bin/python3 venv-rmj
source ~/venvs/venv-rmj/bin/activate
# Install numpy first
pip install numpy
# Install gdal (the correct version)
pip install --global-option=build_ext
↪   --global-option="-I/usr/include/gdal" gdal==$(gdal-config
↪   --version)
pip install riskmapjnr # For PyPI version, this will install all
↪   other dependencies
# pip install
↪   https://github.com/ghislainv/riskmapjnr/archive/master.zip #
↪   For GitHub dev version
# pip install jupyter geopandas descartes folium # Optional
↪   additional packages
```

To deactivate and delete the virtual environment:

```
deactivate
rm -R ~/venvs/venv-rmj # Just remove the repository
```

3. Installation testing

   You can test that the package has been correctly installed using the command `riskmapjnr` in a terminal:

   ```
   riskmapjnr
   ```

   This should return a short description of the `riskmapjnr` package and the version number:

   ```
   # riskmapjnr: Map of deforestation risk following JNR methodology.
   # https://ecology.ghislainv.fr/riskmapjnr/
   # riskmapjnr version x.x.
   ```

   You can also test the package executing the commands in the Get started tutorial of the `riskmapjnr` website.

2.2 Importing Python modules

We import the Python modules needed for running the analysis.

```
# Imports
import os
import multiprocessing as mp
import pkg_resources

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from tabulate import tabulate

import riskmapjnr as rmj
```

Increase the cache for GDAL to increase computational speed.

```
# GDAL
os.environ["GDAL_CACHEMAX"] = "1024"
```

Set the `PROJ_LIB` environmental variable.

```
os.environ["PROJ_LIB"] = os.path.join(
    "/home/ghislain/.pyenv/versions/miniconda3-latest/",
    "envs/conda-rmj/share/proj")
```

Create a directory to save the results.

```
out_dir = "outputs"
rmj.make_dir(out_dir)
```

2.3 Forest cover change data

We use the Guadeloupe archipelago as a case study. Recent forest cover change data for Guadeloupe is included in the `riskmapjnr` package. The raster file (`fcc123_GLP.tif`) includes the following values: **1** for deforestation in the period 2000–2010, **2** for deforestation in the period 2010–2020, and **3** for the remaining forest in 2020. NoData value is set to **0**. The first period (2000–2010) will be used for calibration and the second period (2010–2020) will be used for validation. This is the only data we need to derive a map of deforestation risk following the JNR methodology.

```
fcc_file = pkg_resources.resource_filename(
    "riskmapjnr",
    "data/fcc123_GLP.tif")
print(fcc_file)
border_file = pkg_resources.resource_filename(
    "riskmapjnr",
    "data/ctry_border_GLP.gpkg")
print(border_file)
```

We plot the forest cover change map with the `plot.fcc123()` function.

```
ofile = os.path.join(out_dir, "fcc123.png")
fig_fcc123 = rmj.plot.fcc123(
    input_fcc_raster=fcc_file,
    maxpixels=1e8,
    output_file=ofile,
    borders=border_file,
    linewidth=0.2,
    figsize=(5, 4), dpi=800)
ofile
```

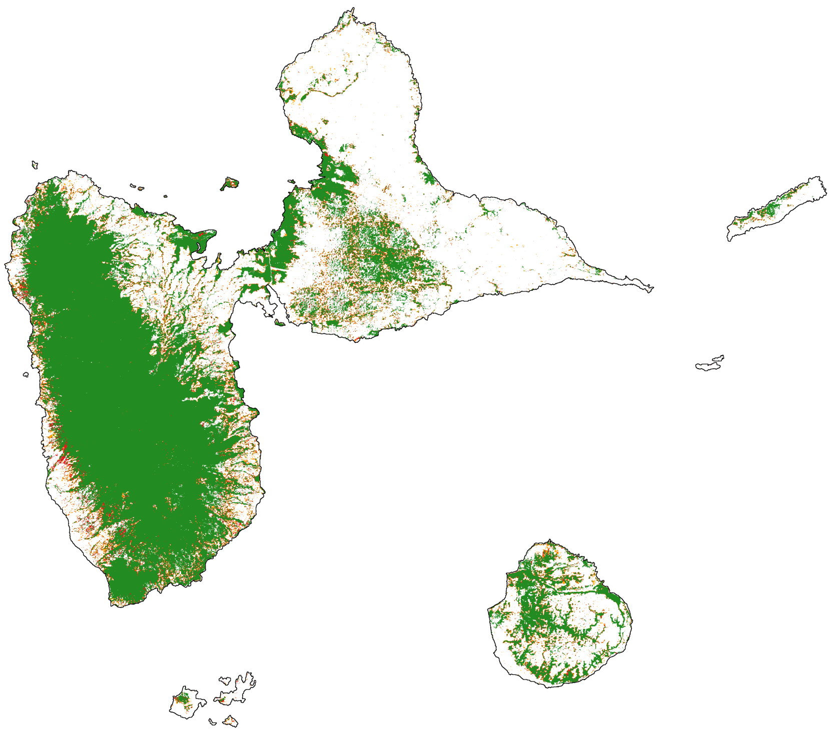

Figure 1: **Forest cover change map.** Deforestation on the first period (2000–2010) is in orange, deforestation on the second period (2000–2020) is in red and remaining forest (in 2020) is in green.

3 Steps to derive a categorical map of deforestation risk

3.1 Deforestation risk with distance to forest edge

The first step is to compute the distance to the forest edge after which the risk of deforestation becomes negligible. Indeed, it is known from previous studies on tropical deforestation that the deforestation risk decreases rapidly with the distance to the forest edge and that most of the deforestation occurs close to the forest edge (Vieilledent et al. 2013, 2022, Dezécache et al. 2017, Grinand et al. 2020). The JNR methodology suggests identifying the distance to the forest edge d, so that at least 99% of the deforestation occurs within a distance $\leq d$. Forest areas located at a distance from the forest edge $> d$ can be considered as having no risk of being deforested. As a consequence, forest pixels with a distance from the forest edge $> d$ are assigned category 0 (zero) for the deforestation risk. The threshold distance can be computed with the function dist_edge_threshold().

```
ofile = os.path.join(out_dir, "perc_dist_steps.png")
dist_edge_thres_steps = rmj.dist_edge_threshold(
    fcc_file=fcc_file,
    defor_values=1,
    dist_file=os.path.join(out_dir, "dist_edge_steps.tif"),
    dist_bins=np.arange(0, 1080, step=30),
    tab_file_dist=os.path.join(out_dir, "tab_dist_steps.csv"),
    fig_file_dist=ofile,
    blk_rows=128, verbose=False)
ofile
```

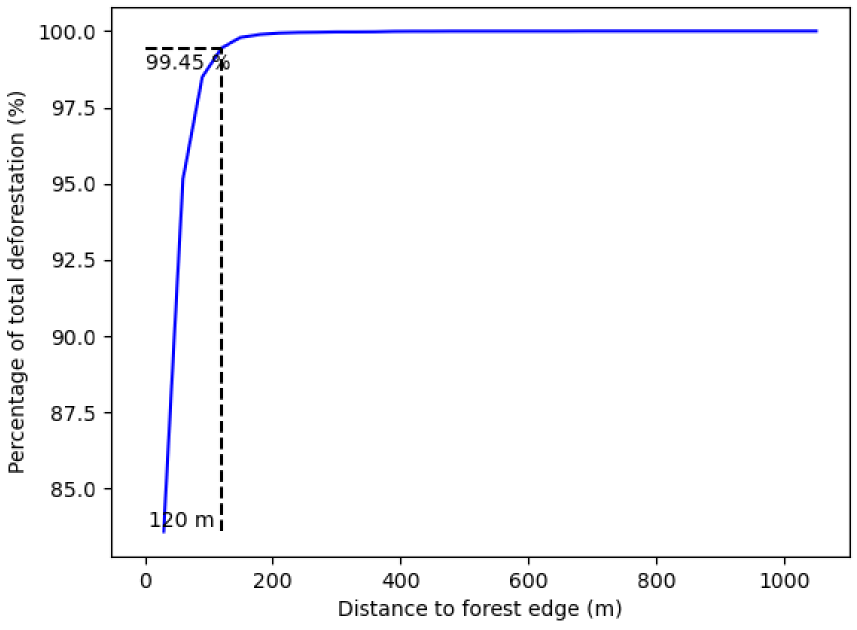

Figure 2: **Identifying areas for which the risk of deforestation is negligible.** Figure shows that more than 99% of the deforestation occurs within a distance from the forest edge ≤ 120 m. Forest areas located at a distance > 120 m from the forest edge can be considered as having no risk of being deforested.

The function returns a dictionary including the distance threshold:

```
dist_thresh = dist_edge_thres_steps["dist_thresh"]
print(f"The distance threshold is {dist_thresh} m")
```

```
The distance threshold is 120 m
```

A table indicating the cumulative percentage of deforestation as a function of the distance is also produced:

Distance	Npixels	Area	Cumulation	Percentage
30	24937	2244.33	2244.33	83.583
60	3451	310.59	2554.92	95.15
90	1001	90.09	2645.01	98.5051
120	282	25.38	2670.39	99.4503
150	102	9.18	2679.57	99.7922
180	29	2.61	2682.18	99.8894
210	14	1.26	2683.44	99.9363
240	6	0.54	2683.98	99.9564
270	2	0.18	2684.16	99.9631
300	3	0.27	2684.43	99.9732

3.2 Local deforestation rate

The second step computes a local risk of deforestation at the pixel level using a moving window made of several pixels. The deforestation risk is estimated from the deforestation rate inside the moving window. The deforestation rate θ (in %/yr) is computed from the formula $\theta = 1 - (\alpha_2/\alpha_1)^{1/\tau}$, with α the forest areas (in ha) at time t_1 and t_2, and τ, the time interval (in yr) between time t_1 and t_2. Using the deforestation rate formula, the moving window and the past forest cover change map, we can derive a raster map describing the local risk of deforestation at the same resolution as the input map. To save space on disk, deforestation rates are converted to integer values between 0 and 10000 (ten thousand) and the raster type is set to UInt16. This ensures a precision of 10^{-4} for the deforestation rate which is sufficient to determine the 30 categories of deforestation risk, as imposed by the JNR methodology.

```python
# Set window size
s = 21
# Compute local deforestation rate
rmj.local_defor_rate(
    fcc_file=fcc_file,
    defor_values=1,
    ldefrate_file=os.path.join(out_dir, f"ldefrate_steps.tif"),
    win_size=s,
    time_interval=10,
    blk_rows=100,
    verbose=False)
```

3.3 Pixels with zero risk of deforestation

This third step sets a value of 10001 to pixels with zero deforestation risk. As explained previously, a risk of deforestation of zero is assumed when distance to forest edge is greater than the distance below which more than 99% of the deforestation occurs.

```
rmj.set_defor_cat_zero(
    ldefrate_file=os.path.join(out_dir, f"ldefrate_steps.tif"),
    dist_file=os.path.join(out_dir, "dist_edge_steps.tif"),
    dist_thresh=dist_thresh,
    ldefrate_with_zero_file=os.path.join(
        out_dir, f"ldefrate_with_zero_steps.tif"),
    blk_rows=128,
    verbose=False)
```

3.4 Categories of deforestation risk

The fourth step implies converting the continuous values of the raster map of deforestation risk to categorical values. The JNR methodology suggests to use 31 classes of risk from “0” to “30” including the “0” class for the forest pixels with no risk of being deforested (located at a distance to the forest edge $> d$, see first step). Following the JNR methodology, at least three slicing algorithms must be compared to derive the categorical map of deforestation risk, such as “equal area”, “equal interval”, and “natural breaks”. With the “equal area” algorithm, each class from “1” to “30” must cover approximately the same area. With the “equal interval” algorithm, classes from “1” to “30” correspond to bins of deforestation risk of the same range. In this case, some risk classes will be in majority in the landscape compared to other classes of lower frequency. With the “natural breaks” algorithm, the continuous deforestation risk is normalized before running an “equal interval” algorithm. To perform this step, we use the function `defor_cat()` which returns the corresponding bins of deforestation probability.

```
bins = rmj.defor_cat(
    ldefrate_with_zero_file=os.path.join(
        out_dir, f"ldefrate_with_zero_steps.tif"),
    riskmap_file=os.path.join(out_dir, "riskmap_steps.tif"),
    ncat=30,
    method="Equal Area",
    blk_rows=128,
    verbose=False)
```

The risk map can be plotted using the `plot.riskmap()` function.

```
ofile = os.path.join(out_dir, "riskmap_steps.png")
riskmap_fig = rmj.plot.riskmap(
    input_risk_map=os.path.join(out_dir, "riskmap_steps.tif"),
    maxpixels=1e8,
    output_file=ofile,
    borders=border_file,
    legend=True,
    figsize=(5, 4), dpi=800,
    linewidth=0.2,)
ofile
```

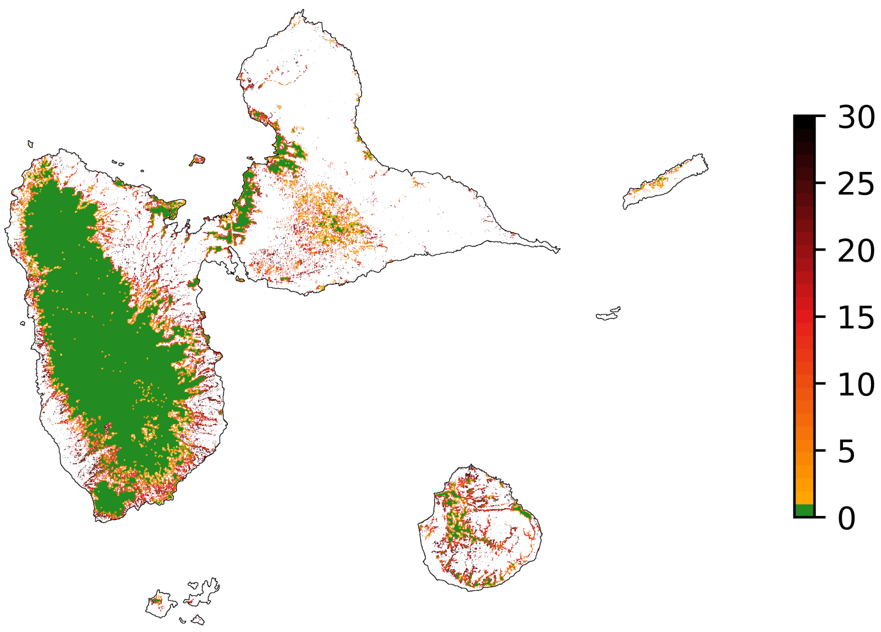

Figure 3: **Map of the deforestation risk following the JNR methodology**. Forest pixels are categorized in up to 30 classes of deforestation risk. Forest pixels which belong to the class 0 (in green) are located farther than a distance of 120 m from the forest edge and have a negligible risk of being deforested.

3.5 Deforestation rates per category of risk

Before the validation step, we need to compute the historical deforestation rates (in %/yr) for each category of spatial deforestation risk. The historical deforestation rates are computed for the calibration period (here 2000–2010). Deforestation rates provide estimates of the percentage of forest (which is then converted to an area of forest) that should be deforested inside each forest pixel which belongs to a given category of deforestation risk.

```
rmj.defrate_per_cat(
    fcc_file=fcc_file,
    defor_values=1,
    riskmap_file=os.path.join(out_dir, "riskmap_steps.tif"),
    time_interval=10,
    tab_file_defrate=os.path.join(out_dir,
     ↪  "defrate_per_cat_steps.csv"),
    blk_rows=128,
    verbose=False)
```

A table indicating the deforestation rate per category of deforestation is produced:

cat	nfor	ndefor	rate
1	70257	0	0
2	17189	47	0.000273768
3	4258	15	0.000352838
4	15311	92	0.000602506
5	17683	110	0.000623815
6	11130	105	0.000947425
7	14920	177	0.00119271
8	12727	164	0.00129613
9	14891	265	0.00179401
10	13132	299	0.00230055
11	14465	426	0.00298481
12	13178	465	0.00358592
13	13776	528	0.00390051
14	14171	670	0.00483168
15	13476	665	0.00504783
16	12955	740	0.00586445
17	13664	1032	0.00782238
18	13135	1101	0.00871624
19	13244	1383	0.0109683
20	13504	1520	0.0118703
21	13482	1969	0.0156639
22	13273	2491	0.0205709
23	13219	3053	0.0259188
24	13364	4382	0.0389551
25	13171	7915	0.0877714

From this table, we see that the deforestation rate increases with the deforestation risk category which is expected. We also see that the "Equal Area" slicing algorithm provides categories with similar forest cover at the beginning of the period (see similar values in the "nfor" column).

3.6 Derive a risk map at the beginning of the validation period

To derive the risk map at the beginning of the validation period, we consider (i) the forest cover at this date, (ii) the map of local deforestation rates, (ii) the threshold distance, and (iii) the bins used to categorize the deforestation rates. All these data are obtained from previous steps and based on the deforestation for the historical period. The approach is the following: first, we consider the forest cover at the beginning of the validation period. Second, we assign category zero to pixels at a distance from the forest edge which is greater than the distance threshold. Third, we categorize the deforestation rates using the previous bins identified for the historical period. In addition to the risk map, two additional raster files are produced: the raster file of the distance to forest edge at the beginning of the validation period, and the raster file of local deforestation rates including the zero deforestation risk.

1. Distance to forest edge at the beginning of the validation period

```
rmj.dist_values(input_file=fcc_file,
                dist_file=os.path.join(out_dir,
                 ↪  "dist_edge_v_steps.tif"),
                values="0,1",
                verbose=False)
```

2. Raster of local deforestation rate at the beginning of the validation period

```
rmj.get_ldefz_v(
    ldefrate_file=os.path.join(out_dir, "ldefrate_steps.tif"),
    dist_v_file=os.path.join(out_dir, "dist_edge_v_steps.tif"),
    dist_thresh=120,
    ldefrate_with_zero_v_file=os.path.join(out_dir,
     ↪  "ldefrate_with_zero_v_steps.tif"),
    blk_rows=128,
    verbose=False)
```

3. Risk map at the beginning of the validation period

```
rmj.get_riskmap_v(
    ldefrate_with_zero_v_file=os.path.join(out_dir,
     ↪  "ldefrate_with_zero_v_steps.tif"),
    bins=bins,
    riskmap_v_file=os.path.join(out_dir, "riskmap_v_steps.tif"),
    blk_rows=128,
    verbose=False)
```

4 Map validation

4.1 Without correcting for quantity disagreement

The fifth step focuses on comparing the map of deforestation risk with a deforestation map corresponding to the validation period. The validation period follows the calibration period and provides independent observations of deforestation. To do so, we consider a square grid of at least 1000 spatial cells containing at least one forest pixel at the beginning of the validation period. Following JNR specifications, the cell size should be $\leq$ 10 km. Note that with the map of deforestation risk, each forest pixel at the beginning of the validation period falls into a category of deforestation risk. For each cell of the grid, we compute the predicted area of deforestation (in ha) given the map of deforestation risk and the historical deforestation rates for each category of deforestation risk computed on the calibration period (see previous step). We can then compare the predicted deforestation with the observed deforestation in that spatial cell for the validation period. Because all cells don't have the same forest cover at the beginning of the validation period, a weight w_j is computed for each grid cell j such that $w_j = \beta_j/B$, with β_j the forest cover (in ha) in the cell j at the beginning of the validation period and B the total forest cover in the jurisdiction (in ha) at the same date. We then calculate the weighted root mean squared error (wRMSE) from the observed and predicted deforestation for each cell and the cell weights.

```
ofile = os.path.join(out_dir, "pred_obs_steps.png")
rmj.validation(
    fcc_file=fcc_file,
    time_interval=10,
    riskmap_file=os.path.join(out_dir, "riskmap_v_steps.tif"),
    tab_file_defrate=os.path.join(out_dir,
     ↪  "defrate_per_cat_steps.csv"),
    csize=40,
    tab_file_pred=os.path.join(out_dir, "pred_obs_steps.csv"),
    fig_file_pred=ofile,
    figsize=(6.4, 4.8),
    dpi=100, verbose=False)
ofile
```

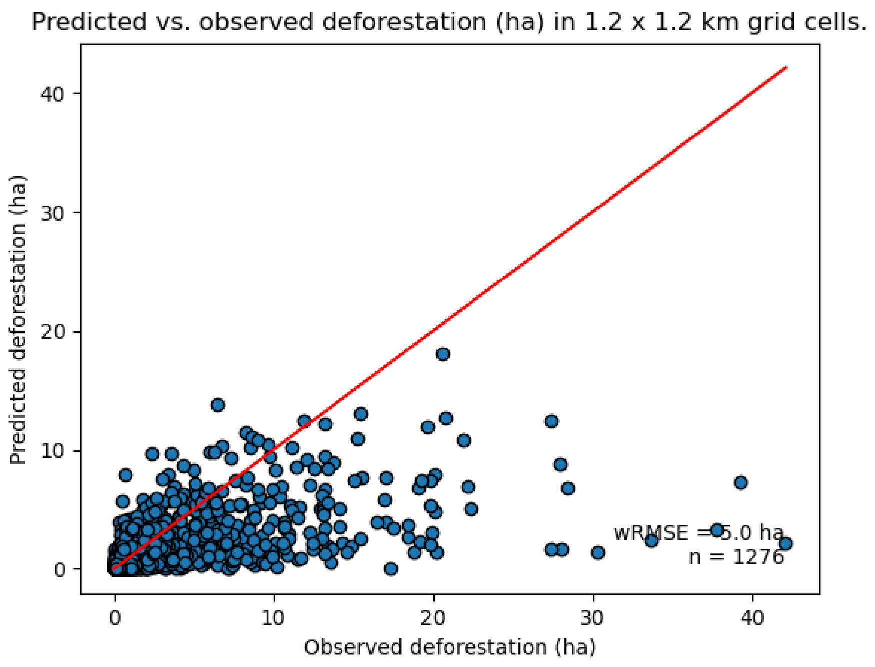

Figure 4: **Relationship between observed and predicted deforestation in 1 x 1 km grid cells**. The red line is the identity line. Values of the weighted root mean squared error (wRMSE, in ha) and of the number of observations (n, the number of spatial cells) are reported on the graph.

4.2 Correcting for quantity disagreement

We set the argument `no_quantity_error` to `True` to correct the total deforestation for the predictions and avoid a "quantity" error (*sensu* Pontius) due to the difference in total deforestation between the calibration and validation periods. This is currently being discussed for improving the JNR methodology.

```
ofile = os.path.join(out_dir, "pred_obs_corrected_steps.png")
rmj.validation(
    fcc_file=fcc_file,
    time_interval=10,
    riskmap_file=os.path.join(out_dir, "riskmap_v_steps.tif"),
    tab_file_defrate=os.path.join(out_dir,
     ↪  "defrate_per_cat_steps.csv"),
    csize=40,
    no_quantity_error=True,
    tab_file_pred=os.path.join(out_dir, "pred_obs_corrected_step.csv"),
    fig_file_pred=ofile,
    figsize=(6.4, 4.8),
    dpi=100, verbose=False)
ofile
```

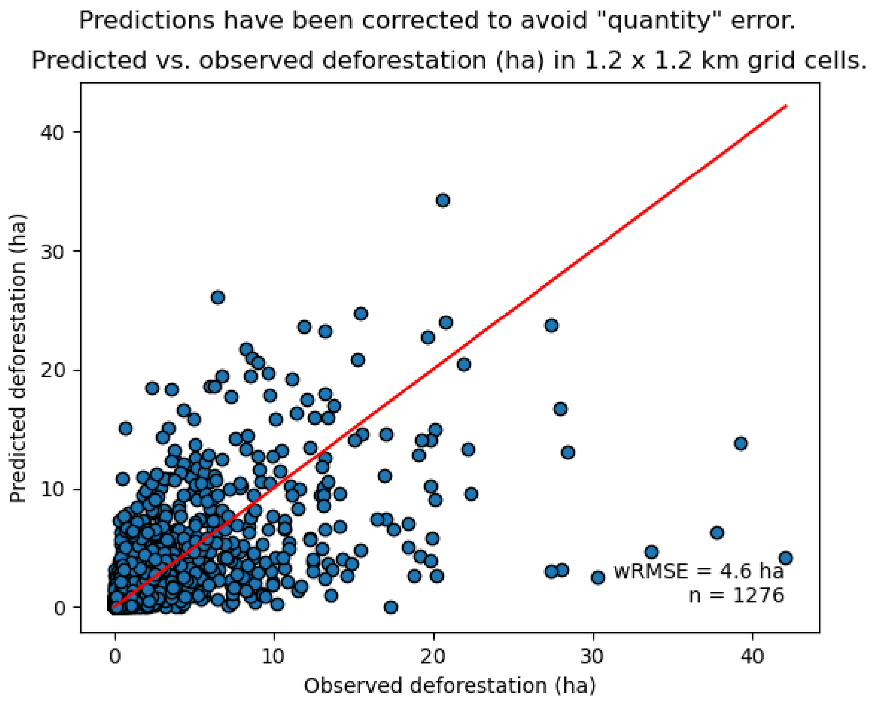

Figure 5: **Relationship between observed and predicted deforestation in 1 x 1 km grid cells *after correction for quantity disagreement*.** The red line is the identity line. Values of the weighted root mean squared error (wRMSE, in ha) and of the number of observations (n, the number of spatial cells) are reported on the graph.

5 Final risk map

5.1 Obtaining several categorical maps with the `makemap` function

The user must repeat the procedure detailed above and obtain risk maps for varying window size and slicing algorithms. Following the JNR methodology, at least 25 different sizes for the moving window must be tested together with two slicing algorithms ("Equal Interval" and "Equal Area"), thus leading to a minimum of 50 different maps of deforestation risk. The map with the smallest wRMSE value is considered the best risk map. Once the best risk map is identified, with the corresponding window size and slicing algorithm, a final risk map is derived considering both the calibration and validation period. The `makemap()` function can be used to run all the procedure with varying window size and slicing algorithm. This function calls a sequence of functions from the `riskmapjnr` package which perform all the steps detailed in the JNR methodology. The function can use parallel computing on several CPUs to reduce computation time. In this case, each map is obtained and validated on one CPU.

```
ncpu = mp.cpu_count() - 2
print(f"Number of CPUs to use: {ncpu}.")
```

```
Number of CPUs to use: 6.
```

```
results_makemap = rmj.makemap(
    fcc_file=fcc_file,
    time_interval=[10, 10],
    output_dir=out_dir,
    clean=False,
    dist_bins=np.arange(0, 1080, step=30),
    win_sizes=np.arange(5, 295, 16),
    ncat=30,
    parallel=True,
    ncpu=ncpu,
    methods=["Equal Interval", "Equal Area"],
    csize=40,
    no_quantity_error=True,
    figsize=(6.4, 4.8),
    dpi=100,
    blk_rows=300,  # Set blk_rows > win_size to avoid error
    verbose=True)
```

```
Model calibration and validation
.. Model 2: window size = 21, slicing method = ei.
.. Model 6: window size = 53, slicing method = ei.
.. Model 0: window size = 5, slicing method = ei.
.. Model 4: window size = 37, slicing method = ei.
.. Model 8: window size = 69, slicing method = ei.
.. Model 10: window size = 85, slicing method = ei.
.. Model 3: window size = 21, slicing method = ea.
.. Model 7: window size = 53, slicing method = ea.
.. Model 1: window size = 5, slicing method = ea.
.. Model 9: window size = 69, slicing method = ea.
.. Model 5: window size = 37, slicing method = ea.
.. Model 11: window size = 85, slicing method = ea.
.. Model 12: window size = 101, slicing method = ei.
.. Model 14: window size = 117, slicing method = ei.
.. Model 16: window size = 133, slicing method = ei.
.. Model 18: window size = 149, slicing method = ei.
.. Model 20: window size = 165, slicing method = ei.
.. Model 22: window size = 181, slicing method = ei.
.. Model 13: window size = 101, slicing method = ea.
.. Model 15: window size = 117, slicing method = ea.
.. Model 21: window size = 165, slicing method = ea.
.. Model 23: window size = 181, slicing method = ea.
.. Model 17: window size = 133, slicing method = ea.
.. Model 19: window size = 149, slicing method = ea.
.. Model 24: window size = 197, slicing method = ei.
.. Model 26: window size = 213, slicing method = ei.
.. Model 30: window size = 245, slicing method = ei.
.. Model 28: window size = 229, slicing method = ei.
.. Model 32: window size = 261, slicing method = ei.
.. Model 34: window size = 277, slicing method = ei.
.. Model 25: window size = 197, slicing method = ea.
.. Model 27: window size = 213, slicing method = ea.
.. Model 31: window size = 245, slicing method = ea.
.. Model 29: window size = 229, slicing method = ea.
.. Model 33: window size = 261, slicing method = ea.
.. Model 35: window size = 277, slicing method = ea.
.. Model 36: window size = 293, slicing method = ei.
.. Model 37: window size = 293, slicing method = ea.
Deriving risk map for full historical period
```

5.2 Updated deforestation risk with distance to forest edge

We obtain the threshold for the distance to forest edge beyond which the deforestation risk is negligible. Because the time period has changed (now we consider the entire period from 2000 to 2020), the distance threshold has increased from 120 to 180 m.

```python
dist_thresh = results_makemap["dist_thresh"]
print(f"The distance theshold is {dist_thresh} m.")
```

```
The distance theshold is 180 m.
```

We have access to the table indicating the cumulative percentage of deforestation as a function of the distance to forest edge.

We also have access to a plot showing how the cumulative percentage of deforestation increases with the distance to forest edge.

```python
os.path.join(out_dir, "fullhist/perc_dist.png")
```

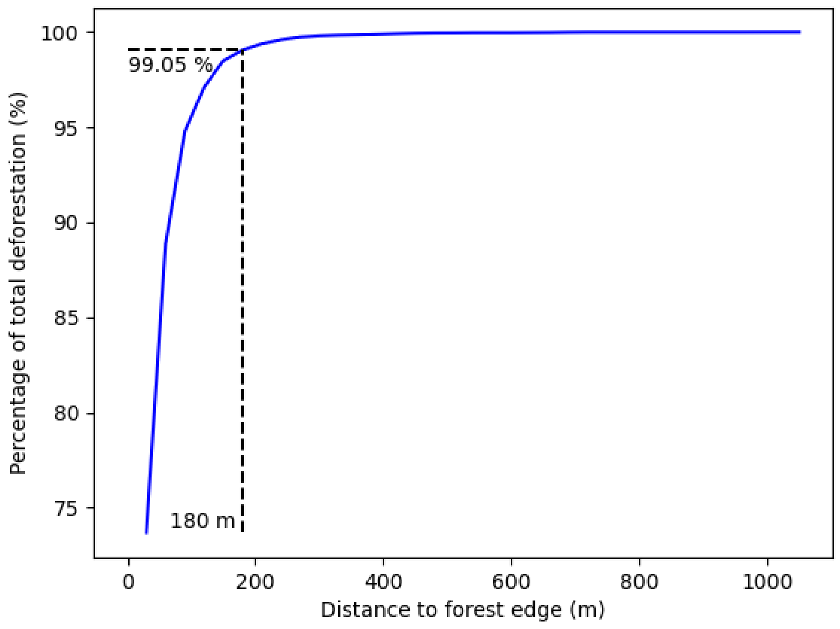

Figure 6: **Identifying areas for which the risk of deforestation is negligible.** Figure shows that more than 99% of the deforestation occurs within a distance from the forest edge ≤ 180 m. Forest areas located at a distance > 180 m from the forest edge can be considered as having no risk of being deforested.

5.3 Model comparison

We can plot the change in wRMSE value with both the window size and slicing algorithm. It seems that the "Equal Interval" (ei) algorithm provides lower wRMSE values. The lowest wRMSE value is obtained for the lowest window size.

```
os.path.join(out_dir, f"modcomp/mod_comp.png")
```

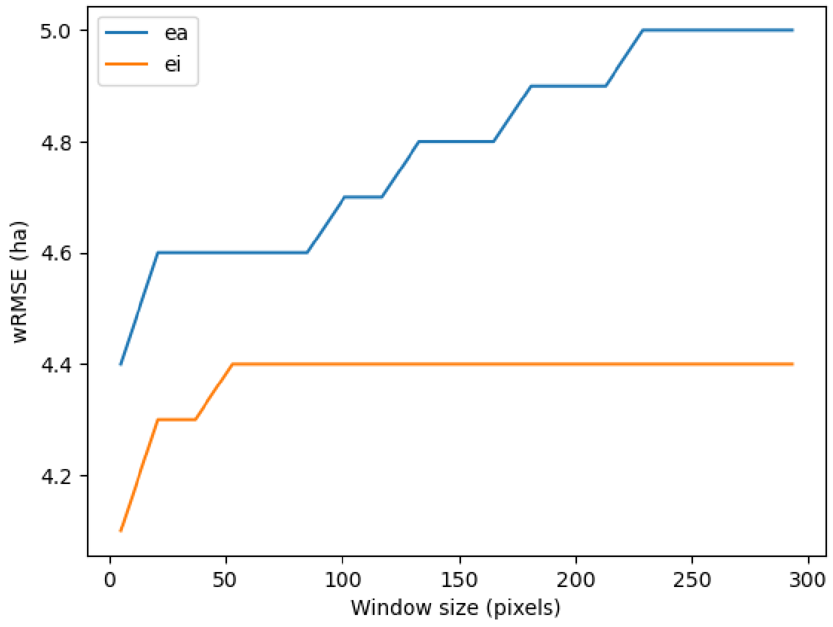

Figure 7: **Change in wRMSE values as a function of both window size and slicing algorithm**. "ei" is the "Equal Interval" algorithm and "ea" is the "Equal Area" algorithm.

We identify the moving window size and the slicing algorithm of the best model.

```
ws_hat = results_makemap["ws_hat"]
m_hat = results_makemap["m_hat"]
print(f"The best moving window size is {ws_hat} pixels.")
print(f"The best slicing algorithm is '{m_hat}'.")
```

```
The best moving window size is 5 pixels.
The best slicing algorithm is 'ei'.
```

5.4 Model performance

We can look at the relationship between observed and predicted deforestation in 1 x 1 km grid cells for the best model.

```
os.path.join(out_dir, f"modcomp/pred_obs_ws{ws_hat}_{m_hat}.png")
```

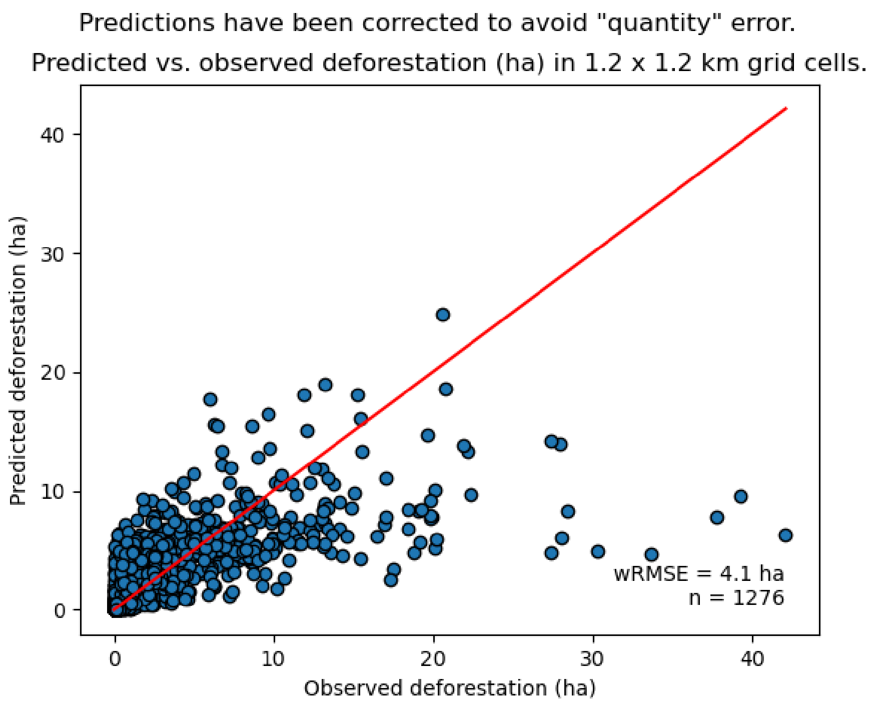

Figure 8: **Relationship between observed and predicted deforestation in 1 x 1 km grid cells for the best model**. The red line is the identity line. Values of the weighted root mean squared error (wRMSE, in ha) and of the number of observations (n, the number of spatial cells) are reported on the graph.

5.5 Risk map of deforestation

We plot the final risk map using the `plot.riskmap()` function.

```
ifile = os.path.join(out_dir,
↪  f"endval/riskmap_ws{ws_hat}_{m_hat}_ev.tif")
ofile = os.path.join(out_dir,
↪  f"endval/riskmap_ws{ws_hat}_{m_hat}_ev.png")
riskmap_fig = rmj.plot.riskmap(
    input_risk_map=ifile,
    maxpixels=1e8,
    output_file=ofile,
    borders=border_file,
    legend=True,
    figsize=(5, 4), dpi=800, linewidth=0.2,)
ofile
```

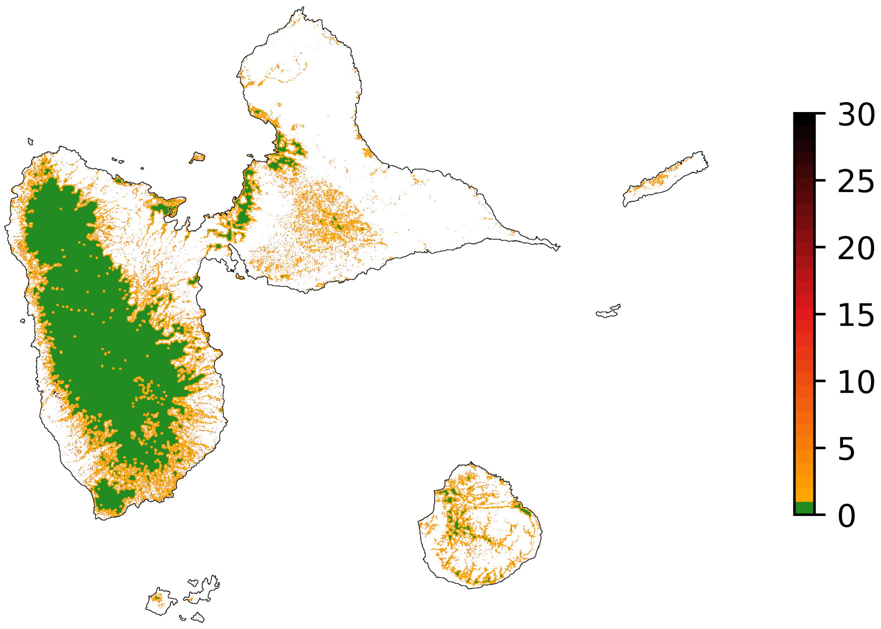

Figure 9: **Map of the deforestation risk following the JNR methodology**. Forest pixels are categorized in up to 30 classes of deforestation risk. Forest pixels which belong to the class 0 (in green) are located farther than a distance of 180 m from the forest edge and have a negligible risk of being deforested.

6 Conclusion

We have presented the use of the `riskmapjnr` Python package to obtain a map of the deforestation risk following the JNR methodology. As said previously, the JNR methodology is currently being discussed and will likely be subject to changes. The first results we have obtained applying the JNR methodology to a small-scale study area allow us to make some recommendations regarding how the JNR methodology could be improved to obtain an accurate map of the deforestation risk. First, the distance threshold above which deforestation become negligible is changing between the calibration period and the whole period (combining the calibration and validation periods) for the final risk map. This is unavoidable as the deforestation generally progresses towards the forest core and the whole period to derive the final map is longer than the calibration period. Here we used the second distance threshold value of 180 m identified using forest cover change data for the whole period 2000–2020. The JNR methodology should be more precise regarding the distance threshold which should be used. Second, it seems that the "Equal Area" slicing algorithm always provides higher wRMSE values than compared with the "Equal Interval" algorithm (Fig. 7). As, a consequence, it might be useless to compute maps using this slicing algorithm. This result must be confirmed with other case studies. Third, there is a large underestimation of the total amount of deforestation for the validation period (Fig. 8). To be able to properly compare deforestation risk maps (each map providing a different way of allocating deforestation spatially *sensu* Pontius and Millones (2011)) there should be no quantity error between deforestation predictions and observations (*sensu* Pontius and Millones (2011)). We suggest not mixing quantity and allocation of deforestation for predictions while the methodology suggests to estimate historical deforestation rates for each category of risk from the calibration period. On the contrary, for the prediction, we suggest to fix the intensity of the deforestation to the one observed on the validation period and use the risk map to identify pixels which should be deforested first.

7 References

Dezécache, C., J.-M. Salles, G. Vieilledent, and B. Hérault. 2017. Moving forward socio-economically focused models of deforestation. Global change biology 23:3484–3500.

Grinand, C., G. Vieilledent, T. Razafimbelo, J.-R. Rakotoarijaona, M. Nourtier, and M. Bernoux. 2020. Landscape-scale spatial modelling of deforestation, land degradation, and regeneration using machine learning tools. Land degradation & development 31:1699–1712.

Pontius, R. G. J., and M. Millones. 2011. Death to Kappa: birth of quantity disagreement and allocation disagreement for accuracy assessment. International journal of remote sensing 32:4407–4429.

Vieilledent, G., C. Grinand, and R. Vaudry. 2013. Forecasting deforestation and carbon emissions in tropical developing countries facing demographic expansion: a case study in Madagascar. Ecology and evolution 3:1702–1716.

Vieilledent, G., C. Vancutsem, C. Bourgoin, P. Ploton, P. Verley, and F. Achard. 2022. Spatial scenario of tropical deforestation and carbon emissions for the 21st century. Biorxiv.