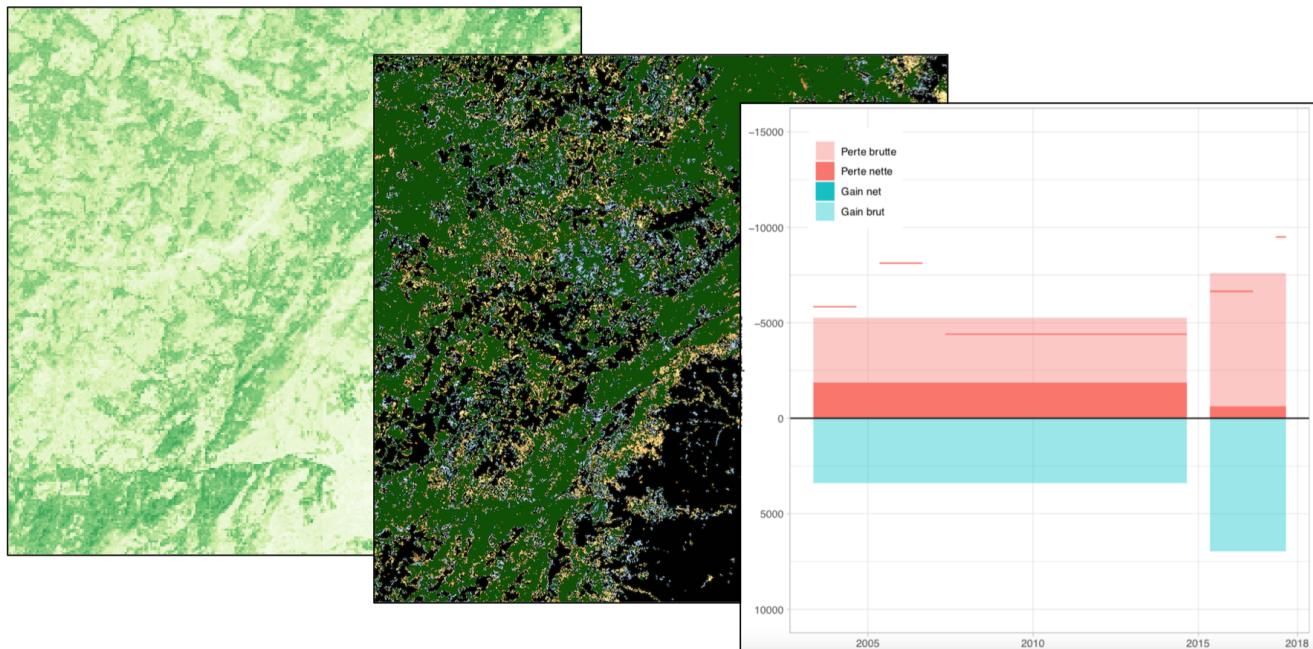


République Togolaise — Système Nationale de Surveillance des Forêts



Préface

Ce manuel de référence à comme objectif de décrire le fonctionnement du **Système National de Surveillance des Forêts au Togo (SNSF)**. Les éléments traités sont les arrangements institutionnelles, l'implémentation de l'Inventaires Forestier National (IFN) et de Système Surveillance Terrestres par Satellite (SSTS) et l'approche technique pour en sortir les informations nécessaires pour le Niveau de Référence pour les Forêts du Togo (NRF) ainsi que pour le Monitoring, Reporting et Verification (MRV) dans le cadre de l'engagement du Togo pour le REDD+.

La partie Analyses NRF/MRV décrit en détail les outils utilisés pour établir et le **Niveau de Référence pour les Forêts du Togo 1.0**, soumis au secrétariat CCNUCC en Janvier 2020 et pour mettre à jour les analyses dans le cadre d'une surveillance de la biomasse forestier continue dans le cadre du Monitoring, Reporting et Verification pour la REDD+. Les résultats de ces analyses sont publiés ailleurs (liens sur les rapports sur le site CCNUCC et géoportail).

En cas de questions, veuillez contacter la Coordination nationale REDD+ du Togo

0.1 Introduction

L'objectif du Système National de Surveillance des Forêts (SNSF) est d'évaluer régulièrement **l'état des forêts togolaises et leur évolution**. Dans le code forestier du Togo, la forêt est définie comme:

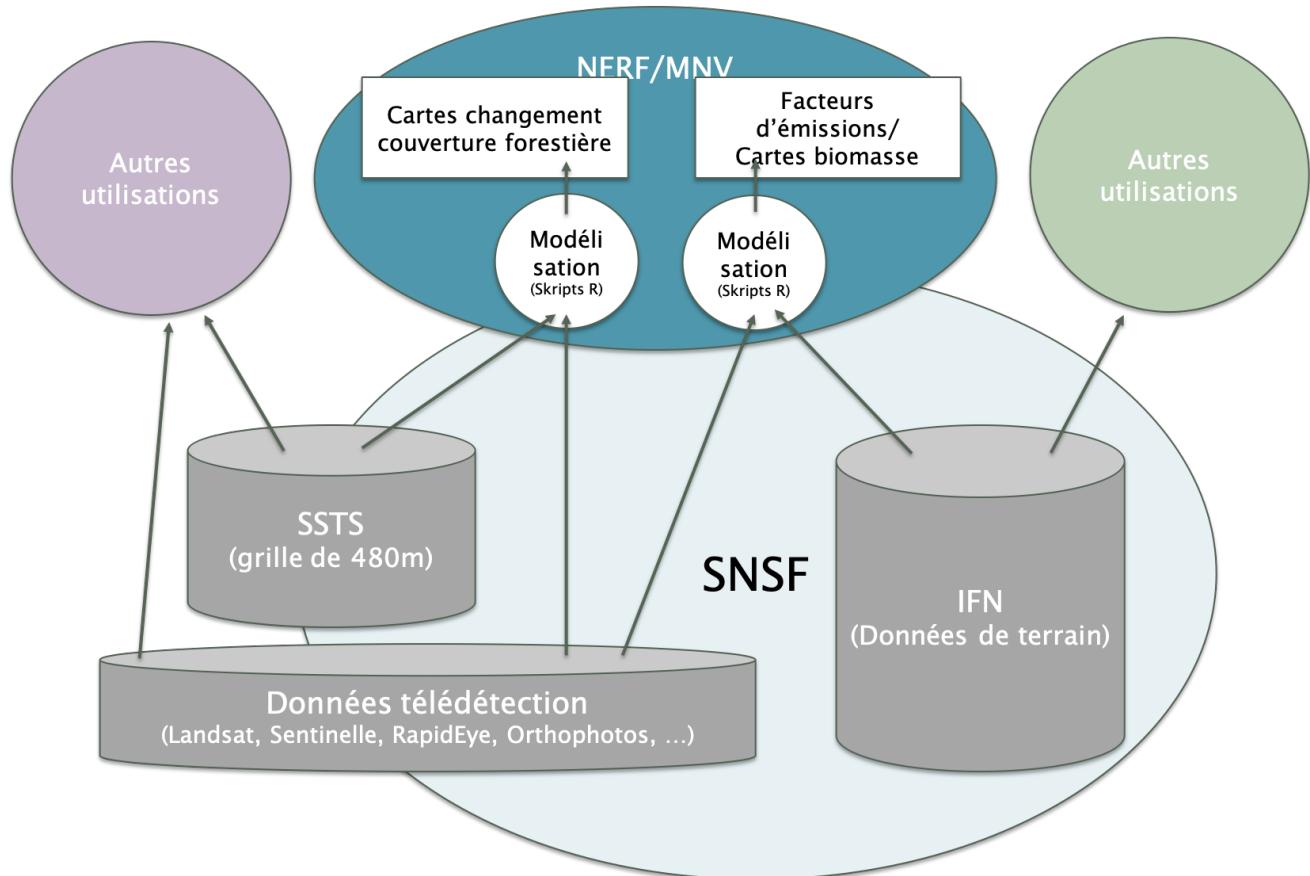
un espace occupant une superficie de plus de 0,5 hectare avec des arbres atteignant une hauteur supérieure à 5 mètres et un couvert arboré de plus de 10 pour cent, ou avec des arbres capables d'atteindre ces seuils in situ.

Pour l'évaluation du développement des zones forestières, le SNSF distingue entre **terres forestières** avec un couvert des houppiers $\geq 30\%$ et les **terres boisées** avec un couvert des houppiers entre $10\% - 30\%$.

 Actuellement, sur base des images Landsat, le SNSF enregistre que l'évolution des terres forestiers avec une couverture des houppiers $\geq 30\%$. Des données satellitaires de plus haute résolution seront nécessaires pour évaluer également les terres boisées.

Le SNSF combine les données recueillies sur le terrain avec les données des images satellites pour fournir des informations sur l'évolution de l'ensemble des forêts dans le pays. Comme illustré dans l'image au-dessous et décrit dans les sections suivantes, **le SNSF consiste de trois piliers principaux**:

- Le **Inventaire Forestier National (IFN)** recueille des informations détaillées sur l'état des forêts sur un nombre limité de placettes d'échantillonnage permanentes sur le terrain.
- Au moyen du **Système de Surveillance Terrestre par Satellite (SSTS)**, des informations sur la couverture et l'utilisation des sols sont recueillies sur un grand nombre de parcelles d'échantillonnage à partir d'images satellites. Avec l'aide du SSTS, les informations sur l'IFN peuvent être extrapolées à l'ensemble du pays.
- Le **Niveau de Référence des Forêts (NRF) ainsi que le Monitoring, Reporting et Vérification (MRV)** des changements dans les réservoirs carbone forestiers est une application du SNSF pour informer la communauté internationale sur l'engagement du Togo dans le cadre du mécanisme REDD+. Dans ce cadre, les données de l'IFN sont utilisées pour déterminer le stockage du carbone dans la biomasse des arbres, tandis que les données du SSTS sont utilisées pour déterminer le changement de la superficie forestière. Ensemble, cela se traduit par les pertes de carbone dues à la déforestation et la séquestration du carbone provenant du reboisement.



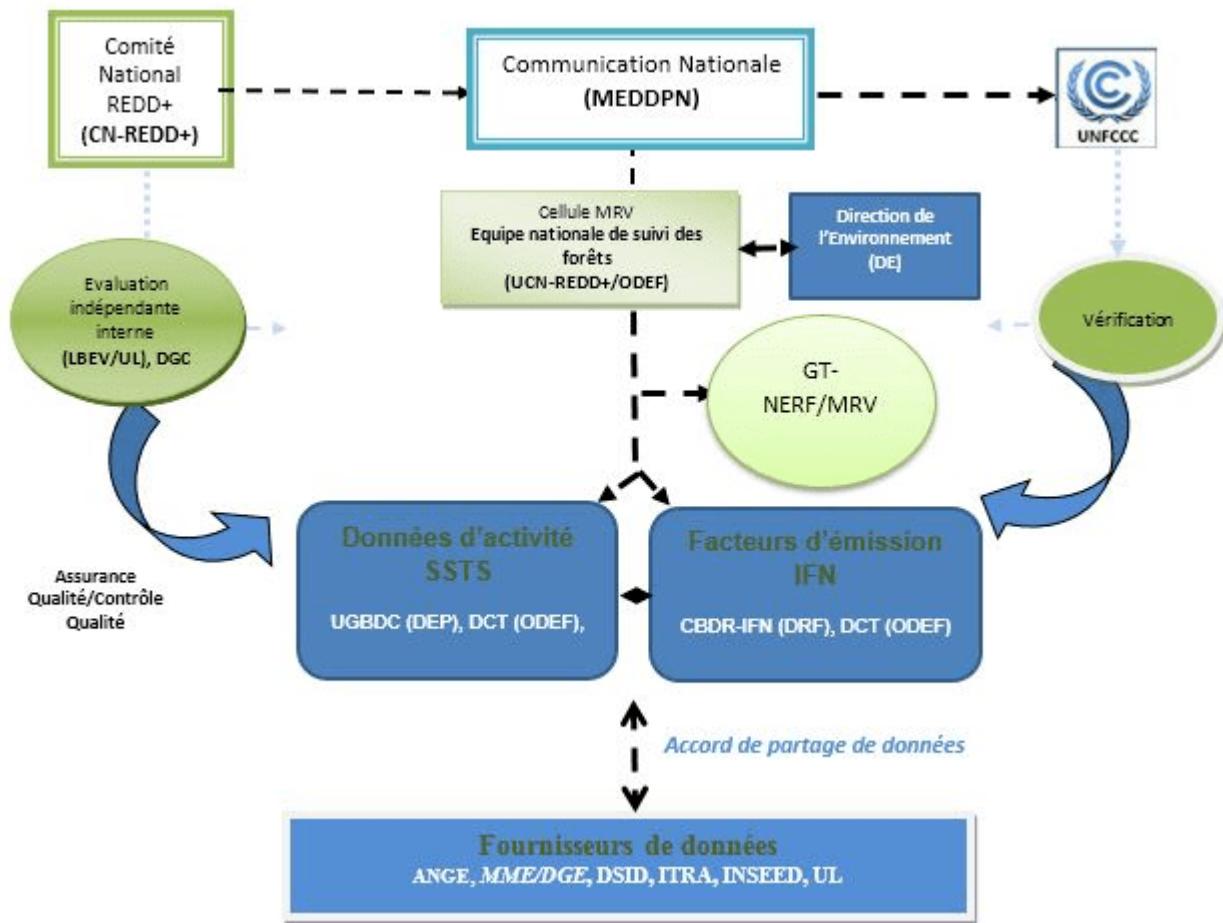
 En future, d'autres sources de données pourraient être intégrées au SNSF, telles que

- le carbone organique du sol (prévu à relever dans l'IFN-2)
- feux de brousse (base de données SANGE)
- dégradation des forêts (utilisation des images de haute résolution Sentinel-2)
- droits fonciers
- plantations
- exploitation du bois
- ...

Pour l'avenir, il est également prévu d'impliquer la population locale dans la surveillance des forêts, par exemple en signalant les activités irrégulières à l'aide d'une application pour smartphone.

0.1.1 Arrangements institutionnelles

L'arrangement institutionnel proposé pour le système national de suivi des forêts se présente comme ci-dessous.



• Coordination:

- Le Ministère de l’Environnement, du Développement Durable et de la Protection de la Nature (MEDDPN) à travers la **Direction de l’Environnement (DE)** est chargée de la **soumission des rapports** (Communication Nationale et Rapports Biennaux) à la Convention Cadre des Nations Unies sur le Changement Climatique (CCNUCC).
- La **Cellule MRV** de la Coordination nationale REDD+ située à l’ODEF est responsable de la **coordination** de toutes les institutions et organisations impliquées dans l’alimentation du système SNSF. Cette cellule est l’entité clé chargée de faciliter et de soutenir les communications sur NRF/NERF du Togo.
- Le **Groupe de travail NERF/MRV** et l’équipe nationale de suivi des forêts sont chargés du travail et des **décisions et choix** techniques sur les données, résultats et méthodologie adoptés pour le NRF/MRV. C’est la cheville ouvrière de la cellule MRV. Elles sont constituées des cadres des institutions qui interviennent dans le système national de suivi des forêts (SNSF).
- La **Direction de l’Environnement (DE)** se charge des **inventaires de gaz à effet de serre (I-GES)** de tous les secteurs mais assure la cohérence des données d’I-GES du secteur agriculture, foresterie et autres affectations des terres (AFAT) avec les rapports qui seront soumis à la CCNUCC. La DE se chargera d’assurer la cohérence entre la méthodologie utilisée dans le cadre du NRF avec les données d’I-GES du secteur AFAT.

• Données d’activités:

- L’**Unité de gestion de bases de données cartographiques (UGBDC)** de la Direction des études et de la planification (DEP), chargée de la gestion de la cartographie

des domaines forestiers du Togo ainsi que la **Division cartographie et Télédétection (DCT)** de l'Officie de développement et d'exploitation des forêts (ODEF) chargée de la cartographie des forêts classées et plantations étatique se chargeront de produire les **données d'activités à travers le système de suivi des terres par satellite (SSTS)**.

- L'Agence nationale de gestion de l'environnement (ANGE) est chargée de fournir les **données sur les feux de végétation**.

- Facteurs d'émission:

- La **cellule de gestion de la base des données des ressources forestières et des résultats de l'inventaire forestier national (CBDR-IFN)** de la Direction des ressources forestières (DRF) et la **Division cartographie et télédétection (DCT)** de l'ODEF sont chargé de produire les facteurs d'émission à travers les **inventaires forestier nationaux et les inventaires des plantations**.

- Données complémentaires:

- la **Direction générale de l'énergie du ministère des mines et énergie DGE/MME** se chargera de fournir les données sur la **consommation en bois énergie**.
- la **Direction de la Statistique agricole de l'Informatique et de la Documentation (DCID)** et l'**Institut Togolais de Recherche Agronomique (ITRA)** produiront des **données sur l'agriculture (superficie emblavées et le cheptel)**.
- les **données de recherche des universités du Togo** alimenteront le mécanisme MRV ainsi que le NRF.
- L'**Institut national de la statistique et des études économiques et démographiques (INSEED)** donnera des compléments d'informations sur la **démographie et autres**.

- Contrôle de qualité / validation interne:

- L'assurance qualité et le contrôle qualité se fera à travers l'évaluation indépendante interne du **Laboratoire de biologie et écologie végétale (LBEV)** et le **Laboratoire de recherche forestière (LRF)** de l'université de Lomé (LBEV/UL) ainsi que la **Direction générale de la cartographie (DGC)**. Les laboratoires universitaires LRF et LBEV évalueront les méthodes et nouveaux données au fur et à mesure qu'ils seront générés.

0.1.2 Pour commencer

0.1.2.1 Travailler avec le SNSF

0.1.2.1.1 Installation R et RStudio

Le traitement des données au sein du SNSF est principalement basé sur les scripts R (*R Project for Statistical Computing*). Pour pouvoir travailler avec ces scripts, vous devez disposer d'une installation de R. En outre, l'installation des paquets suivants est nécessaire dans R :

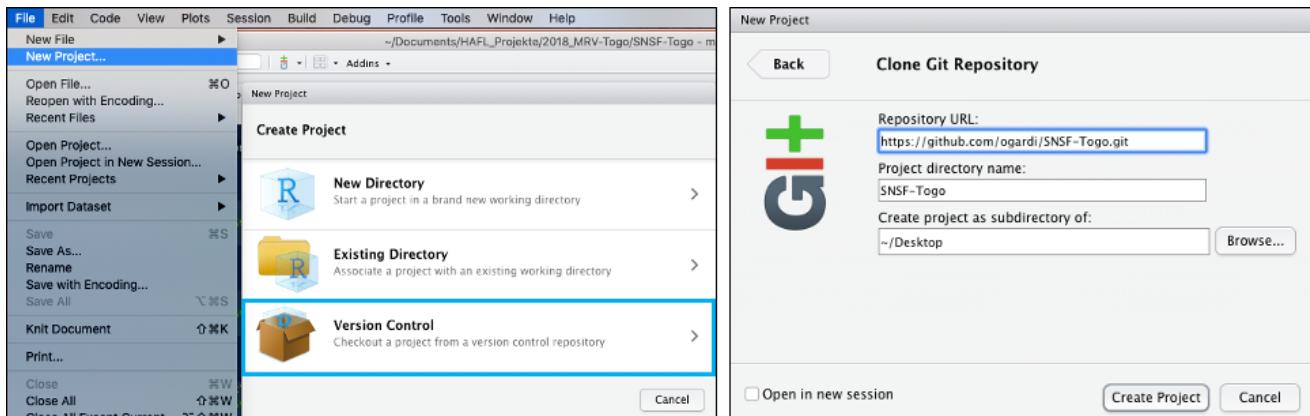
```
install.packages("sp", "rgdal", "raster", "randomForest", "caret", "openxlsx",
                 "dplyr", "tidyverse", "ggplot2", "foreach", "doParallel", "knitr",
                 "rmarkdown", "tinytex")
```

Les scripts R dépendent des fois des outils GDAL disponible dans l'environnement. Sur les systèmes Linux, ceux-ci peuvent être installés en utilisant ‘apt-get install python-gdal’.

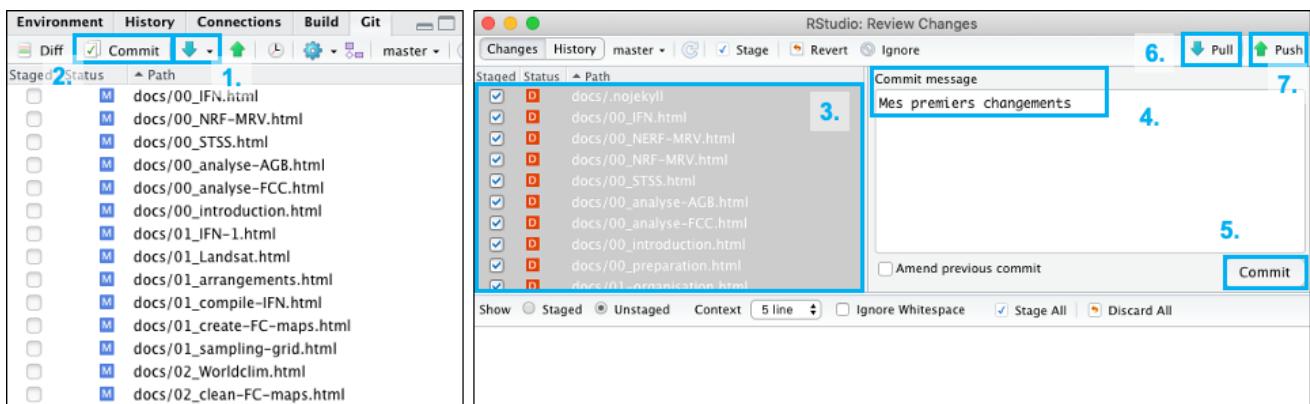
Comme interface utilisateur graphique pour R, nous recommandons l'installation de RStudio.

0.1.2.1.2 Se connecter avec GitHub

Le code SNSF-Togo se trouve sur un dépôt du code GitHub (<https://github.com/ogardi/SNSF-Togo>). Pour copier le code sur votre ordinateur, vous pouvez dans RStudio créer un nouveau projet à partir d'un dépôt Git comme illustré dans la figure ci-dessous.



Git vous permet de vous synchroniser avec le dépôt sur GitHub. En appuyant sur le bouton “Pull” (point 1. dans la figure ci-dessous), vous mettez à jour votre code local avec le dépôt du code sur GitHub.



Pour modifier vous-même le code sur GitHub, vous devez disposer d'un compte GitHub et d'une autorisation du gestionnaire du dépôt. La procédure à suivre pour apporter des modifications au code est la suivante.

Avant faire des modifications dans le code:

1. synchroniser avec le dépôt sur GitHub en utilisant le bouton “Pull” (tirer)

De temps en temps, pendant le travail:

2. confirmer les modifications apportées au code en appuyant sur le bouton “Commit”. Une nouvelle fenêtre s’ouvrira.
3. sélectionner les fichiers à confirmer (CTRL-A) et les marquer (ESPACE)
4. joindre un message (ce qui a été modifié)
5. confirmer les modifications apportées au code en appuyant sur le bouton “Commit” (confirmer)
6. synchroniser de nouveau avec GitHub (et résoudre les éventuels conflits)
7. télécharger les modifications sur GitHub en utilisant le bouton “Push” (pousser)

0.1.2.1.3 Travailler sur la documentation

Ce manuel de référence fait également partie du dépôt du code sur GitHub. Cela signifie que pour réviser la documentation, vous devez procéder comme décrit ci-dessus.

La documentation a été créée dans Markdown, une syntaxe simple pour la structuration et le formatage des documents. Ces documents Markdown sont ensuite traduits en HTML par le paquet R `knitr` pour générer un site web.

```
---
title: "Example"
author: "Oliver Gardi"
date: "5/18/2020"
output: html_document
---

# Titre
## Sous-titre

normal

*italic*

**gras**

* bullet 1
* bullet 2
* bullet 3

`code`

| Tableau | col 1 | col 2 |
|:-----|:-----|-----|
| ligne 1 | abc | 123 |
| ligne 2 | def | 456 |

[Markdown sur Wikipedia](https://fr.wikipedia.org/wiki/Markdown)

![Logo Markdown](https://upload.wikimedia.org/wikipedia/commons/thumb/4/48/Markdown-mark.svg/130px-Markdown-mark.svg.png)
```



Example
Oliver Gardi
5/18/2020

Titre

Sous-titre

normal

italic

gras

- bullet 1
- bullet 2
- bullet 3

code

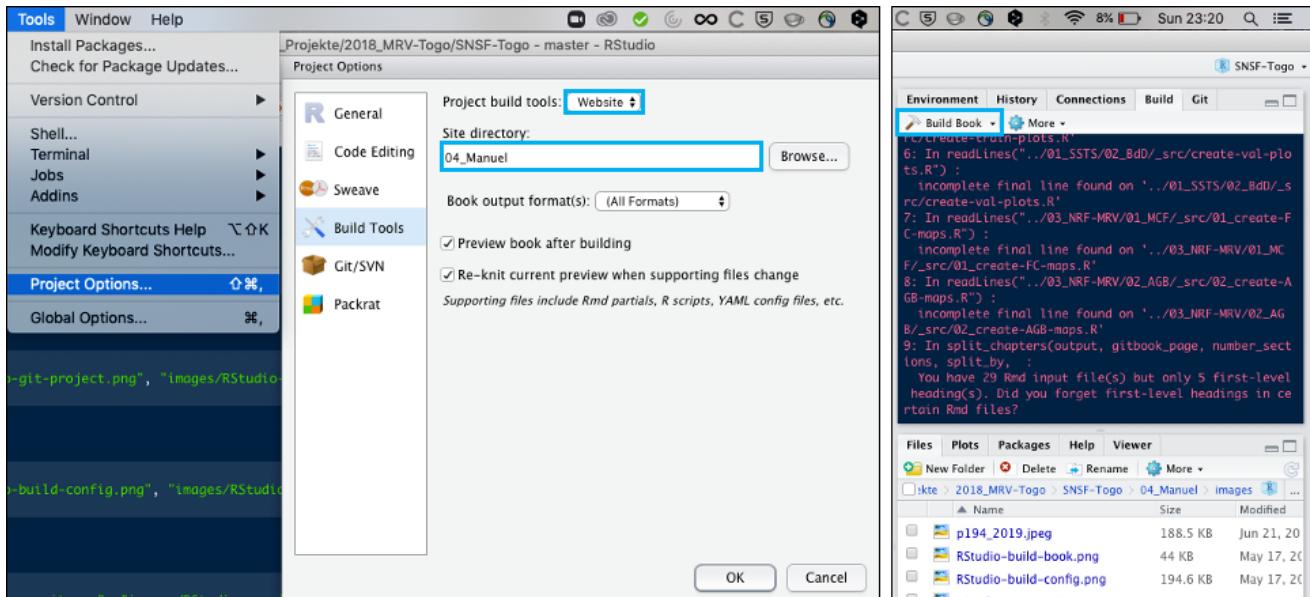
Tableau	col 1	col 2
ligne 1	abc	123
ligne 2	def	456

[Markdown sur Wikipedia](#)

Logo Markdown

Le répertoire `04_Manuel` contient les documents Markdown (fichiers .Rmd), le répertoire `docs` contient le site web résultant (fichiers .html).

Dans RStudio, le site web peut être généré en utilisant le bouton “Build Book”. Pour ce faire, il faut d’abord spécifier le répertoire dans lequel se trouvent les fichiers Markdown dans les configurations, comme indiqué dans la figure ci-dessous.



0.1.2.2 Structure du répertoire

Les outils R dépendent d'une certaine structure des fichiers. Le répertoire de base SNSF_Togo est structuré comme suivant:

```

SNSF_Togo
=====
|--- data
    |--- GADM
    |--- Landsat
    |--- SRTM
    |--- Worldclim
# Données de base externes #####
# :: frontières administratives
# :: images satellitaires
# :: données topographiques
# :: données climatiques

|_-- SNSF_v1.0_20200106
    |--- .Rprofile
# Répertoire SNSF v1.0 #####
#::: Script R de initialisation

    |--- 01_SSTS
        |--- 01_data
        |--- 02_BdD
# SYSTÈME DE SURVEILLANCE TERRESTRE =====
#::: images et autres données pré-traités --
#::: base de données SSTS --

    |--- 02_IFN
        |--- 01_IFN-1
# INVENTAIRE FORESTIER NATIONAL =====
#::: données d'inventaire IFN-1 --

    |--- 03_NRF-MRV
        |--- 01_MCF
        |--- 02_AGB
        |--- 03_report
# NIVEAU DE REFERENCE / MRV =====
#::: Modification Couvert Forestier --
#::: cartographie biomasse --
#::: rapport NRF/MRV --

    |--- 04_Manuel
    |--- docs
# CETTE DOCUMENTATION DU SNSF =====
#::: site web manuel de référence

|_-- SNSF_v1.x
    |--- ...
    |--- ...
# Répertoire SNSF version actualisé #####

```

La structure du répertoire est définit dans le script R `.Rprofile` et peut être ajouté.

C'est seulement les répertoires `src` et `manual` qui sont mis à disposition dans le dépôt GitHub. Les autres répertoires et données doivent être installés manuellement.

0.1.2.3 Crédit d'un nouveau projet

Pour la création d'un nouveau projet avec le code le plus actuel, on clone le dépôt du projet sur GitHub par la commande `git clone --single-branch https://github.com/ogardi/NERF-Togo.git NOM-REPERTOIRE`. L'installation d'une version spécifique peut être fait avec `git clone -b VERSION --single-branch https://github.com/ogardi/NERF-Togo.git NOM-REPERTOIRE`.

En travaillant avec RStudio, le plus facile est de créer directement un projet RStudio au départ du dépôt GitHub (`File > New Project ... > Version Control > Git`) avec les mêmes paramètres que l'installation directe en haut.

Dans une prochaine étape, les données doivent être rendues disponibles, soit par une nouvelle acquisition, soit par une copie des répertoires existants.

- S'il n'est pas encore disponible, le répertoire `../data/` doit être créé et les données de base correspondantes doivent être fournies.
- Dans le répertoire `./01_SSTS/02_BdD/`, le réseau d'échantillonnage ainsi que les données d'entraînement et de validation doivent être stockées.
- En outre, les données d'inventaire doivent être stockées dans le répertoire `./02_IFN/`.

0.1.2.4 Définition des variables

Le traitement des images ainsi que les différentes analyses se font via des R-Scripts. Les variables et les fonctions utilisées dans les différents scripts sont définies dans le fichier `.Rprofile`, qui est automatiquement chargé au démarrage de R. Si le processus de chargement a réussi, vous pouvez voir un message de bienvenue suivant sur la console R.

Si aucun message n'apparaît, assurez-vous que a) R est lancé dans le répertoire et b) qu'aucun message d'erreur ne se produit. Les messages d'erreur possibles sont des paquets manquants (les paquets correspondants doivent être installés en premier) ou le fait de ne pas trouver les limites administratives de GADM dans `../data/GADM` (doivent également être installés en premier).

Si nécessaire les variables sont ajustées et R est redémarré. Notamment les informations sur la période analysée `YEARS.ALL` et les années à prendre en compte pour les différentes évaluations `YEARS.JNT`, `YEARS.VAL` et `YEARS.REF`.

0.1.2.4.1 Script R: `.Rprofile`

```
#####
# .Rprofile: Préparer l'environnement (libraries, variables)
# -----
# Bern University of Applied Sciences
```

```

# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Charger libraries =====
library("sp")           # Classes et méthodes pour les données spatiales
library("rgdal")         # Geospatial Data Abstraction Library
library("raster")        # Analyse et modélisation des données géographiques
library("randomForest")  # Algorithme de classification et régression
library("caret")          # Outils pour classification et régression
library("openxlsx")       # Lire et écrire des fichiers Excel (xlsx)
library("dplyr")          # Fonctions pour manipuler des données
library("tidyverse")      # Fonctions pour reorganiser des données
library("ggplot2")        # Production des figures
library("RColorBrewer")   # palettes de couleurs
library("foreach")        # Faire des calculs en parallèle ...
library("doParallel")     # ... sur plusieurs processeurs
library("knitr")          # pour la documentation html

# Créer un environnement caché =====
.snsf = new.env()

# Années / Périodes -----
.snsf$YEARS.ALL <- 1985:2019                                # - tous
.snsf$YEARS.JNT <- c(1987, 2003, 2005, 2007, 2015, 2017, 2018) # - conjointes
.snsf$YEARS.REF <- c(1987, 2003,                      2015,          2018) # - référence
.snsf$YEARS.NRF <- c(          2003,                      2015,          2018) # - NRF

# Répertoires -----
.snsf$DIR.RAW.DAT    <- "../data"

.snsf$DIR.SST         <- "./01_SSTS"
.snsf$DIR.SST.DAT    <- paste0(.snsf$DIR.SST, "/01_data")
.snsf$DIR.SST.DAT.LST <- paste0(.snsf$DIR.SST.DAT, "/Landsat")
.snsf$DIR.SST.DAT.WC2 <- paste0(.snsf$DIR.SST.DAT, "/Worldclim")

.snsf$DIR.SST.BDD    <- paste0(.snsf$DIR.SST, "/02_BdD")
.snsf$DIR.SST.BDD.GRD <- paste0(.snsf$DIR.SST.BDD, "/01_reseau-SSTS")
.snsf$DIR.SST.BDD.TPS <- paste0(.snsf$DIR.SST.BDD, "/02_train-plots")
.snsf$DIR.SST.BDD.VPS <- paste0(.snsf$DIR.SST.BDD, "/03_val-plots")

.snsf$DIR.IFN         <- "./02_IFN"
.snsf$DIR.IFN.DAT    <- paste0(.snsf$DIR.IFN, "/01_field-data")

.snsf$DIR.MRV         <- "./03_NRF-MRV"
.snsf$DIR.MRV.MCF    <- paste0(.snsf$DIR.MRV, "/01_MCF")
.snsf$DIR.MRV.MCF.REF <- paste0(.snsf$DIR.MRV.MCF, "/01_ref-maps")

```

```

.snsf$DIR.MRV.MCF.RAW    <- paste0(.snsf$DIR.MRV.MCF, "/02_raw-maps")
.snsf$DIR.MRV.MCF.CLN    <- paste0(.snsf$DIR.MRV.MCF, "/03_cln-maps")
.snsf$DIR.MRV.MCF.VAL    <- paste0(.snsf$DIR.MRV.MCF, "/04_validation")

.snsf$DIR.MRV.AGB      <- paste0(.snsf$DIR.MRV, "/02_AGB")
.snsf$DIR.MRV.AGB.REF   <- paste0(.snsf$DIR.MRV.AGB, "/01_ref-maps")
.snsf$DIR.MRV.AGB.RES   <- paste0(.snsf$DIR.MRV.AGB, "/02_results")

# Système de référence des coordonnées ----

 snsf$UTM.30 <- crs("+proj=utm +zone=30 +datum=WGS84
                      +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0")

 snsf$UTM.31 <- crs("+proj=utm +zone=31 +datum=WGS84
                      +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0")

# Domaine d'intérêt ----

.snsf$TGO      <- spTransform(
                     readOGR(paste0(.snsf$DIR.RAW.DAT, "/GADM/gadm36_TGO_0.shp")),
                     .snsf$UTM.31
                   )

snsf$TGO.REG   <- spTransform(
                     readOGR(paste0(.snsf$DIR.RAW.DAT, "/GADM/gadm36_TGO_1.shp")),
                     .snsf$UTM.31
                   )

.snsf$TGO.EXT <- extent(151155, 373005, 670665, 1238175) # xmin, xmax, ymin, ymax

# Noms des couches de données ----

.snsf$SST.LSBANDS     <- c("B", "G", "R", "NIR", "SWIR1", "SWIR2",
                           "evi", "msavi", "nbr", "nbr2", "ndmi", "ndvi", "savi")

.snsf$SST.BIOCLIM     <- paste0("BIO", 1:19)

# Codes pour les classes ----

.snsf$NONFOR <- 0 # non-forêt
.snsf$PREGEN <- 1 # régénération potentielle
.snsf$REGEN  <- 2 # régénération
.snsf$FOREST <- 3 # forêt / forêt initiale

# Divers ----

# Processeurs disponibles pour le calcul parallèle
.snsf$CORES <- detectCores()

# Semence pour le générateur de nombres aléatoires

```

```
.snsf$RSEED <- 20191114

# Attacher l'environnement
attach(.snsf)

# Message de bienvenue =====
message("=> chargé librairies et variables définit en .Rprofile

.oPYo. o     o .oPYo.  ooooo  ooooo
8      8b    8 8      8        8
'Yooo. 8'b   8 'Yooo. o8oo      8     .oPYo. .oPYo. .oPYo.
   '8 8 'b 8   '8 8      8     8 8     8 8     8 8     8
   8 8 'b8    8 8      8     8 8     8 8     8 8     8
'YooP' 8     '8 'YooP' 8       8     'YooP' 'YooP8 'YooP'
:.....:.....:.....:.....:.....:.....:.....:.....:.....:8 :....:
:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:ooP':.....:
:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:.....:
\n",
date(),
"\n"
)
```

0.2 Système de Surveillance Terrestre

Le Système de Surveillance Terrestre par Satellite (SSTS) consiste d'une base de données spatiales (images satellitaires, modèle numérique du terrain, données climatiques) d'un réseau de parcelles d'échantillonnage, dont la **couverture des houppiers et l'utilisation des terres** sont régulièrement enregistrées par photo-interprètes sur base des images satellitaires. Pour le moment, les attributs enregistrés dans le SSTS sont :

- Couverture des houppiers
- Occupation des terres

Les répertoires et les fichiers sont structurés comme suit:

```
01_SSTS                                # SYSTÈME DE SURVEILLANCE TERRESTRE =====
|--- 01_data                            #: images et autres données pré-traités --
|   |-- ...
|--- 02_BdD                             #: base de données SSTS -----
|   |--- _src                            #: scripts R
|       |--- _create-grid.R            # [R] création réseau SSTS
|       |--- _create-train-plots.R    # [R] création parcelles d'entraînement
```

```

|_-- _create-val-plots.R          # [R] création parcelles de validation
|--- 01_reseau-SSTS              #: réseau de parcelles SSTS
|--- 02_train-plots              #: parcelles d'entraînement
|_-- 03_val-plots                #: parcelles de validation

```



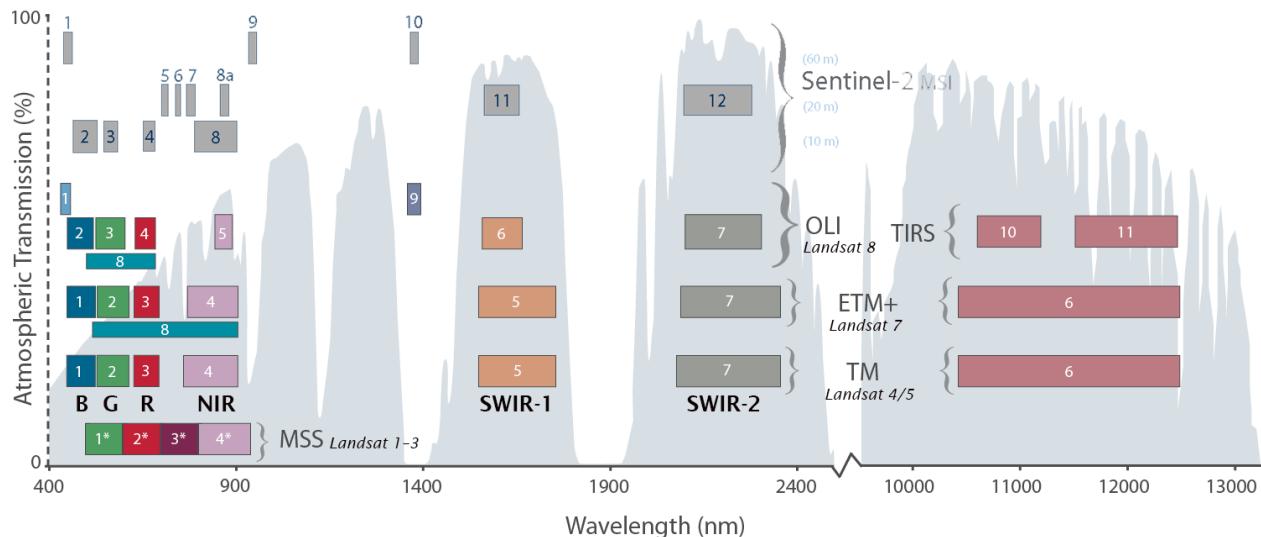
Actuellement, les données de base ainsi que les données générées par les photo-interprètes sont stockées dans des fichiers (tif pour les rasters et shp pour les données vecteurs). Il est prévu de gérer ces données SSTS dans une base de données géographiques (PostGIS) à l'avenir. Cette base de données est actuellement installé sur un serveur central au ministère de l'Environnement et des Forêts (MERF).

0.2.1 Données de base

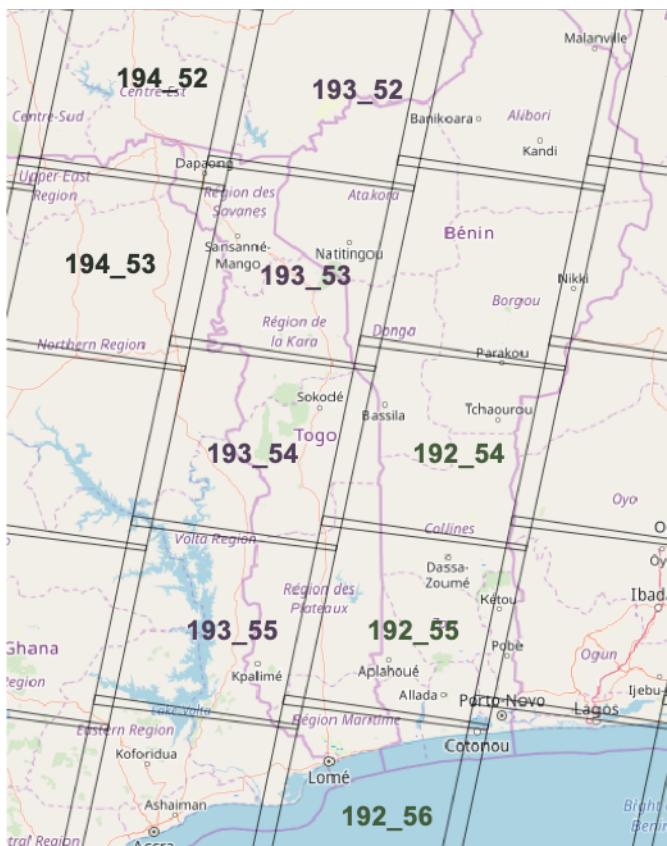
0.2.1.1 Images Landsat

Actuellement, l'analyse de l'évolution du couvert forestier est principalement basée sur les **images satellitaires Landsat**, qui sont disponibles gratuitement dans les archives de l'USGS. Les missions Landsat-4 à Landsat-8 produisent des images de résolution spatiale et radiométrique comparable depuis les années 1980. Les images brutes sont corrigées géométriquement et radiométriquement par l'USGS (USGS Collection 1 Level-2 Surface Reflection Product).

La résolution spatiale des images est de **30 mètres**. Les bandes spectrales utilisées sont **B, G, R, NIR, SWIR-1 et SWIR-2** (voir les désignations des bandes dans la figure ci-dessous).



Le territoire du Togo est couvert par un total de 9 images Landsat (scènes WRS 2). Les zones couvertes par les différentes scènes sont indiquées dans la figure ci-dessous.



Les images Landsat utilisées pour l'évaluation du couvert forestier ont été prises idéalement à la fin de la saison sèche, c'est-à-dire de (Nov), Déc, Jan, (Fév) et sont disponibles en bonne qualité (sans nuages ou seulement légèrement couvertes par des nuages et des ombres) pour la même date sur l'ensemble du chemin WRS 2 respectif. Pour les années de référence, là où on aimerait des cartes qui couvrent l'ensemble du territoire du Togo, des images correspondantes sont nécessaires pour tous les trois chemins WRS 2.

Le tableau ci-dessous présente les images satellites utilisées pour l'analyse du couvert forestier au Togo. Les années de référence utilisées pour le NRF et les images correspondantes sont indiquées en caractères gras. Ce n'est que pour l'année 1991 que les images de différentes dates d'enregistrement ont été combinées pour le chemin WRS 193. L7* marque les images Landsat-7 avec des lacunes dans les données (SLC-off). La colonne GoogleEarth montre la répartition des dates des images de très haute résolution disponibles sur GoogleEarth. Seules les images de référence GoogleEarth de 2017 – 2018 ont été utilisées pour la calibration de la carte forêt/non-forêt 2018.

	WRS 192 ^{054,055,056}	WRS 193 ^{052,053,054,055}	WRS 194 ^{052,053}	GoogleEarth ^{Référence}
2019	L8 / 23.12.18	L8 / 16.02.19	L8 / 22.01.19	++
2018	L8 / 05.01.18	L8 / 12.01.18	L8 / 18.12.17	++++++
2017	L8 / 19.02.17	L8 / 25.01.17	L8 / 31.12.16	+++
2016	—	—	—	(+)
2015	L8 / 13.01.15	L8 / 04.01.15	L8 / 27.01.15	(+)
2014	—	—	—	(++)
2013	L7* / 31.01.13	L7* / 23.02.13	—	(+)
2012	—	—	L7* / 11.01.12	(++)
2011	L7* / 10.01.11	—	—	(+)
2010	—	—	L7* / 21.01.10	(+)
2009	—	L7* / 27.01.09	—	
2008	—	—	—	

	WRS 192 ^{054,055,056}	WRS 193 ^{052,053,054,055}	WRS 194 ^{052,053}	GoogleEarth ^{Référence}
2007	L7* / 30.12.06	L7* / 22.01.07	L5 / 05.01.07	
2006	—	—	—	
2005	L7* / 24.12.04	L7* / 17.02.05	L7* / 22.12.04	
2004	—	—	—	
2003	L7 / 04.01.03	L7 / 26.12.02	L7 / 17.12.02	(.)
2002	—	—	—	
2001	L7 / 13.12.00	—	—	(.)
2000	—	L7 / 04.02.00	L7 / 26.01.00	
...	
1997	—	—	L5 / 10.02.97	
...	
1991	L4 / 03.01.91	L4 / 10.01.91 & L5 / 28.11.89	—	
...	
1987	L5 / 31.12.86	L5 / 23.01.87	L5 / 29.12.86	
1986	L5 / 13.01.86	L5 / 06.03.85	L5 / 11.01.86	

0.2.1.1.1 Acquisition des images

Ouvrir le site USGS Earthexplorer. Dans la fenêtre **Search Criteria** il faut sélectionner la période pour laquelle on cherche des images (Nov - Jan). Dans la fenêtre **Data Sets**, les produits Landsat Level-2 (Surface Reflectance) sont sélectionnés. Dans la fenêtre **Additional Criteria** il faut choisir les scènes (chemin 192: 054-056 / chemin 193: 052-055 / chemin 194: 052-053).

EE EarthExplorer - Home X ESPA - New Bulk Order https://earthexplorer.usgs.gov

Home

Search Criteria **Data Sets** **Additional Criteria** **Results**

1. Enter Search Criteria

To narrow your search area: type in an address or place name, enter coordinates or click the map to define your search area (for advanced map tools, view the [help documentation](#)), and/or choose a date range.

Geocoder **KML/Shapfile Upload**

Select a Geocoding Method
Address/Place

Address/Place Show Clear

Polygon Circle Predefined Area

Degree/Minute/Second Decimal

No coordinates selected.

Use Map Add Coordinate Clear Coordinates

Date Range Result Options

Search from: 11/01/2019 to: 02/24/2020

Search months: (all)

Data Sets » **Additional Criteria** » **Results** »

2. Select Your Data Set(s)

Check the boxes for the data set(s) you want to search. When done selecting data set(s), click the **Additional Criteria** or **Results** buttons below. Click the plus sign next to the category name to show a list of data sets.

Use Data Set Prefilter ([What's This?](#))

Data Set Search:

- Declassified Data
- Digital Elevation
- Digital Line Graphs
- Digital Maps
- EO-1
- Global Fiducials
- HCMM
- ISERV
- Land Cover
- Landsat

 - Landsat Collection 1 Level-3
 - Landsat Analysis Ready Data (ARD)
 - Landsat Collection 1 Level-2 (On-Demand)
 - Landsat 8 OLI/TIRS C1 Level-2
 - Landsat 7 ETM+ C1 Level-2
 - Landsat 4-5 TM C1 Level-2
 - Landsat Collection 1 Level-1
 - Landsat Legacy
 - NASA LPDAC Collections
 - Radar

Clear All Selected Additional Criteria Results »

3. Additional Criteria (Optional)

If you have more than one data sets selected, use the dropdown to select the additional criteria for each data set.

Data Sets: **Landsat 8 OLI/TIRS C1 Level-2**

WRS Path 194 to

WRS Row 052 to 053

Land Cloud Cover All

- Less than 10%
- Less than 20%
- Less than 30%
- Less than 40%

Scene Cloud Cover All

- Less than 10%
- Less than 20%
- Less than 30%
- Less than 40%

Collection Category All

- Tier 1
- Tier 2

Clear All Criteria Results »

EE EarthExplorer - Home X ESPA - New Bulk Order https://earthexplorer.usgs.gov

Home

Search Criteria **Data Sets** **Additional Criteria** **Results**

4. Search Results

If you selected more than one data set to search, use the dropdown to see the search results for each specific data set.

Note: You must be logged in to download and order scenes

Show Result Controls

Data Set Click here to export your results »

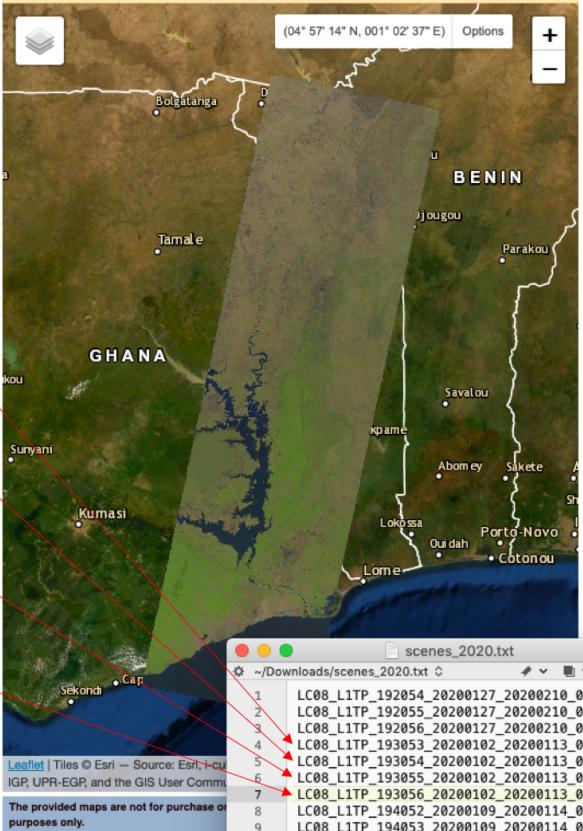
Landsat 8 OLI/TIRS C1 Level-2

17	ID:LC08_L1TP_193053_20200102_20200113_01_T1 Acquisition Date:02-JAN-20 Path:193 Row:53  View Download Order Remove
18	ID:LC08_L1TP_193054_20200102_20200113_01_T1 Acquisition Date:02-JAN-20 Path:193 Row:54  View Download Order Remove
19	ID:LC08_L1TP_193055_20200102_20200113_01_T1 Acquisition Date:02-JAN-20 Path:193 Row:55  View Download Order Remove
20	ID:LC08_L1TP_193056_20200102_20200113_01_T1 Acquisition Date:02-JAN-20 Path:193 Row:56  View Download Order Remove

« First < Previous 2 > Next > Last »

Search Criteria Summary (Show)

(04° 57' 14" N, 001° 02' 37" E) Options



Clear Search Criteria

scenes_2020.txt

```
1 LC08_L1TP_192054_20200127_20200210_01_T1
2 LC08_L1TP_192055_20200127_20200210_01_T1
3 LC08_L1TP_192056_20200127_20200210_01_T1
4 LC08_L1TP_193053_20200102_20200113_01_T1
5 LC08_L1TP_193054_20200102_20200113_01_T1
6 LC08_L1TP_193055_20200102_20200113_01_T1
7 LC08_L1TP_193056_20200102_20200113_01_T1
8 LC08_L1TP_194052_20200109_20200114_01_T1
9 LC08_L1TP_194053_20200109_20200114_01_T1
```

The provided maps are not for purchase or commercial purposes only.

L: 7 C: 41 Text File Unicode (UTF-8)

DOI Privacy Policy | **Legal** | **Accessibility** | **Site Map** | **Contact USGS**

U.S. Department of the Interior | DOI Inspector General | White House | Email | No Fear Act | FOIA

Parmi les images disponibles, on sélectionne celles qui sont disponibles à la même date et en bonne qualité pour l'ensemble du chemin WRS. On copie les identifier des images à télécharger dans un fichier txt.

Ensuite on ouvre le site USGS ESPA pour commander les images choisi. On charge le fichier txt avec les identifier des images et on commande les bandes **Surface Reflectance** et les indices spectrales (voir image au-dessous). Pour commander des images, il faut qu'on a un compte USGS.

NOTICES

Reminders:

- **ESPA users are allowed 10,000 open / in process units at any given time.**
- Data from **ESPA** is available to download for 7 days once orders complete processing.
- Please contact custserv@usgs.gov with any questions.

Add Input Products ([Show Available Products](#))

Scene List

scenes_2020.txt

Select Product Contents

Source Products

Input Products
 Input Product Metadata

Additional Processing (Landsat Only)

Level-2 Products

Surface Reflectance - *Not available for thermal or panchromatic bands*
 Top of Atmosphere Reflectance
 Brightness Temperature - *Thermal band TOA processing*
 Pixel QA
 Spectral Indices

NDVI
 EVI
 SAVI
 MSAVI
 NDMI
 NBR
 NBR2

Additional MODIS/VIIRS Processing

Spectral Indices (NDVI only)

MODIS Daily 500-m NDVI
 VIIRS Daily 500-m NDVI

Customize Outputs

Customization Options

Output Format GeoTiff ENVI HDF-EOS2 NetCDF
 Reproject Products
 Modify Image Extents
 Pixel Resizing

Intercomparison & Statistics

Plot Output Product Statistics

Add Order Description

Order Description (optional)

SNSP Togo 2020

Accessibility **FOIA** **Privacy** **Policies and Notices**

U.S. Department of the Interior U.S. Geological Survey
Page Contact Information: custserv@usgs.gov
Page Last Modified: Wednesday, Oct 25, 2017 at 09:10:36



Une fois on est notifié par eMail que les images sont prêts, on les téléchargent manuellement ou tous ensemble avec le USGS bulkdownloader et la commande `download_espa_order.py -u [nom d'utilisateur] -o ALL -d [répertoire]`. On dézip les images et les rangent dans le répertoire `data/Landsat` sous la scène et l'année correspondante. Pour des images de l'hiver 2019/20, l'année correspondante est 2020.

0.2.1.1.2 Prétraitement des images

Le premier traitement est la préparation des images Landsat et autres variables utilisés pour modéliser la surface forestier ou la biomasse aérienne comme les données topographique et climatiques. L'objectif est qu'on prépare avec les données brutes un jeu de données raster complet sur le même extent (territoir du Togo) et avec la même résolution spatiale de 30 mètres (résolution de base des images Landsat).

On ouvre le script `1_prepare-images.R` et on modifie la liste des images Landsat à utiliser (`in.image.list`), par exemple par ajouter les nouveaux image à considérer dans les analyses:

```
p192.2019 = list(
  paste0(DATA.DIR, "/Landsat/192_054/2019/LC081920542018122301T1-SC20190405164258/"),
  paste0(DATA.DIR, "/Landsat/192_055/2019/LC081920552018122301T1-SC20190405163359/"),
  paste0(DATA.DIR, "/Landsat/192_056/2019/LC081920562018122301T1-SC20190405163342/"))
...
p193.2019 = list(
  paste0(DATA.DIR, "/Landsat/193_052/2019/LC081930522019021601T1-SC20190405183839/"),
  paste0(DATA.DIR, "/Landsat/193_053/2019/LC081930532019021601T1-SC20190405181518/"),
  paste0(DATA.DIR, "/Landsat/193_054/2019/LC081930542019021601T1-SC20190405183609/"),
  paste0(DATA.DIR, "/Landsat/193_055/2019/LC081930552019021601T1-SC20190405181507/"))
...
p194.2019 = list(
  paste0(DATA.DIR, "/Landsat/194_052/2019/LC081940522019012201T1-SC20190405172019/"),
  paste0(DATA.DIR, "/Landsat/194_053/2019/LC081940532019012201T1-SC20190405172055/"))
```

Outre la définition des images à traiter, le script définit une fonction `prepare.image` pour stacker les différentes bandes des images Landsat, pour les fusionner, masquer et couper les images Landsat chemin par chemin (WRS2 paths 192, 193 et 194 pour Togo). Par défaut, les images qui ont déjà été traité (`filename` existe déjà) ne sont plus traité (`overwrite=FALSE`).

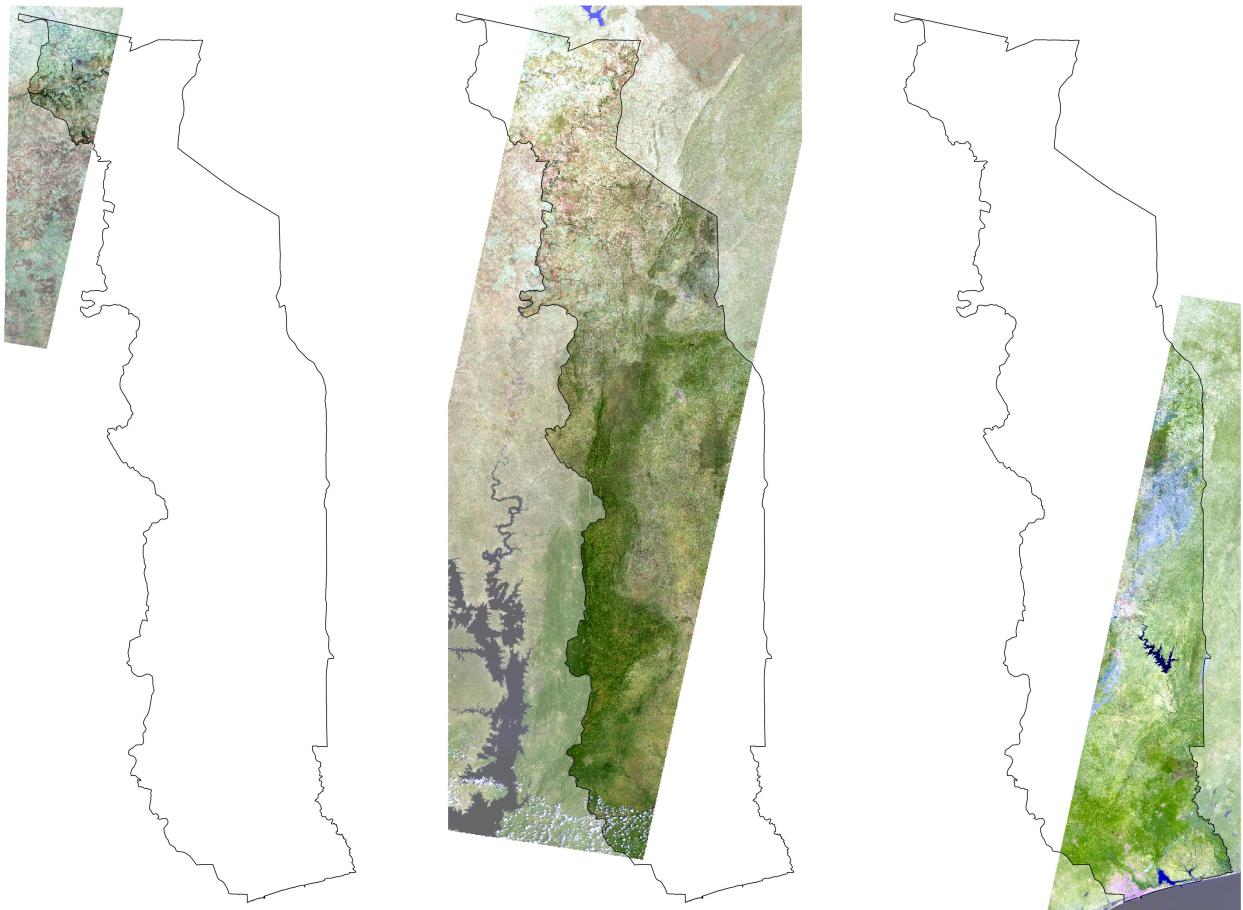
```
prepare.image(in.image.dirs, ext=NULL, filename=NULL, overwrite=FALSE, log=TRUE)
```

Dans la deuxième partie du script, là où c'est noté `# DO THE WORK -----`, on lance le traitement des images. Avec le code `foreach(...) %dopar% { ... }` on lance le traitement de chaque chemin pour chaque année sur des différents processeurs au parallèle. À la fin du script on

- transforme les images du chemin 194 du système de coordonnées UTM 30 vers UTM 31
- produit des thumbnails des images Landsat

Les images prétraités sont sauvegarder dans le répertoire `input/1_images` du projet, ensemble avec des Thumbnails des chemins. Dans une prochaine étape, les images sont nettoyées de l'eau, nuages et ombres en utilisant les bandes Landsat de qualité des pixels.

0.2.1.1.2.1 Example



Images Landsat de l'année 2019: chemin p194 composé de 2 scènes du 22.01.2019 / p193 avec 4 scènes du 16.02.2019 / p192 avec 3 scènes du 23.12.2018

0.2.1.1.3 Script R: 01_SSTS/01_data/_src/prep-Landsat.R

```
#####
# prep-Landsat.R: lire, nettoyer et empiler des images Landsat
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables =====
IN.DIR <- paste0(DIR.RAW.DAT, "/Landsat")

OUT.DIR <- paste0(DIR.SST.DAT, "/Landsat")
if(!dir.exists(OUT.DIR)) dir.create(OUT.DIR)

WRS      <- readOGR(paste0(IN.DIR, "/WRS2/WRS2_descending.shp")) # Scènes WRS
```

```

# Masques de nuages et d'eau pour Landsat 4-7 et Landsat 8 -----
# voir les guides des produits de réflexion de surface Landsat 4-7 et Landsat 8
# https://www.usgs.gov/land-resources/nli/landsat/landsat-surface-reflectance

QA.WATER <- c(
  68, 132,                               # Landsat 4-7
  324, 388, 836, 900, 1348               # Landsat 8
)

QA.SHADOW <- c(
  72, 136,
  328, 392, 840, 904, 1350
)

QA.ICE <- c(
  80, 112, 144, 176,
  336, 368, 400, 432, 848, 880, 912, 944, 1352
)

QA.CLOUD <- c(
  96, 112, 160, 176, 224,
  352, 368, 416, 432, 480, 864, 880, 928, 944, 992
)

# Liste des images à préparer et à fusionner =====

in.image.list <- list(
  # Chemin WRS p192 -----
  p192.1986 = list(paste0(IN.DIR, "/192_054/1986/LT051920541986011301T1-SC20190405164223",
                        paste0(IN.DIR, "/192_055/1986/LT051920551986011301T1-SC20190405164227",
                        paste0(IN.DIR, "/192_056/1986/LT051920561986011301T1-SC20190405164153"))

  p192.1987 = list(paste0(IN.DIR, "/192_054/1987/LT051920541986123101T1-SC20190405164150",
                        paste0(IN.DIR, "/192_055/1987/LT051920551986123101T1-SC20190405163521",
                        paste0(IN.DIR, "/192_056/1987/LT051920561986123101T1-SC20190405164444"))

  p192.1991 = list(paste0(IN.DIR, "/192_054/1991/LT041920541991010301T1-SC20190405164201",
                        paste0(IN.DIR, "/192_055/1991/LT041920551991010301T1-SC20190405165911",
                        paste0(IN.DIR, "/192_056/1991/LT041920561991010301T1-SC20190405163911"))

  p192.2001 = list(paste0(IN.DIR, "/192_054/2001/LE071920542000121301T1-SC20190405165521",
                        paste0(IN.DIR, "/192_055/2001/LE071920552000121301T1-SC20190405165645",
                        paste0(IN.DIR, "/192_056/2001/LE071920562000121301T1-SC20190405164029"))

  p192.2003 = list(paste0(IN.DIR, "/192_054/2003/LE071920542003010401T1-SC20190520111322",
                        paste0(IN.DIR, "/192_055/2003/LE071920552003010401T1-SC20190520100402",
                        paste0(IN.DIR, "/192_056/2003/LE071920562003010401T1-SC20190520100206"))
)

```

```

p192.2005 = list(paste0(IN.DIR, "/192_054/2005/LE071920542004122401T1-SC20190405165520
                     paste0(IN.DIR, "/192_055/2005/LE071920552004122401T1-SC20190405164050
                     paste0(IN.DIR, "/192_056/2005/LE071920562004122401T1-SC20190405164030

p192.2007 = list(paste0(IN.DIR, "/192_054/2007/LE071920542006123001T1-SC20190406034211
                     paste0(IN.DIR, "/192_055/2007/LE071920552006123001T1-SC20190406034231
                     paste0(IN.DIR, "/192_056/2007/LE071920562006123001T1-SC20190406034202

p192.2011 = list(paste0(IN.DIR, "/192_054/2011/LE071920542011011001T1-SC20190406034214
                     paste0(IN.DIR, "/192_055/2011/LE071920552011011001T1-SC20190406034114
                     paste0(IN.DIR, "/192_056/2011/LE071920562011011001T1-SC20190406034155

p192.2013 = list(paste0(IN.DIR, "/192_054/2013/LE071920542013013101T1-SC20190406034224
                     paste0(IN.DIR, "/192_055/2013/LE071920552013013101T1-SC20190406034046
                     paste0(IN.DIR, "/192_056/2013/LE071920562013013101T1-SC20190406034057

p192.2015 = list(paste0(IN.DIR, "/192_054/2015/LC081920542015011301T1-SC20190405163446
                     paste0(IN.DIR, "/192_055/2015/LC081920552015011301T1-SC20190405163723
                     paste0(IN.DIR, "/192_056/2015/LC081920562015011301T1-SC20190405164231

p192.2017 = list(paste0(IN.DIR, "/192_054/2017/LC081920542017021901T1-SC20190405163339
                     paste0(IN.DIR, "/192_055/2017/LC081920552017021901T1-SC20190405163342
                     paste0(IN.DIR, "/192_056/2017/LC081920562017021901T1-SC20190405163222

p192.2018 = list(paste0(IN.DIR, "/192_054/2018/LC081920542018010501T1-SC20190405164304
                     paste0(IN.DIR, "/192_055/2018/LC081920552018010501T1-SC20190405163402
                     paste0(IN.DIR, "/192_056/2018/LC081920562018010501T1-SC20190405163250

p192.2019 = list(paste0(IN.DIR, "/192_054/2019/LC081920542018122301T1-SC20190405164258
                     paste0(IN.DIR, "/192_055/2019/LC081920552018122301T1-SC20190405163359
                     paste0(IN.DIR, "/192_056/2019/LC081920562018122301T1-SC20190405163342

# Chemin WRS p193 -----
p193.1985 = list(paste0(IN.DIR, "/193_052/1985/LT051930521985030601T1-SC20190520100259
                     paste0(IN.DIR, "/193_053/1985/LT051930531985030601T1-SC20190520100324
                     paste0(IN.DIR, "/193_054/1985/LT051930541985030601T1-SC20190520100340
                     paste0(IN.DIR, "/193_055/1985/LT051930551985030601T1-SC20190520100140

p193.1987 = list(paste0(IN.DIR, "/193_052/1987/LT051930521987012301T1-SC20190405182322
                     paste0(IN.DIR, "/193_053/1987/LT051930531987012301T1-SC20190405182335
                     paste0(IN.DIR, "/193_054/1987/LT051930541987012301T1-SC20190405182331
                     paste0(IN.DIR, "/193_055/1987/LT051930551987012301T1-SC20190405182328

# Attention: là nous avons des images avec des dates différentes !
p193.1990.1 = list(paste0(IN.DIR, "/193_052/1990/LT051930521989112801T1-SC201905201002
                     paste0(IN.DIR, "/193_053/1990/LT051930531989112801T1-SC201905201002
p193.1990.2 = list(paste0(IN.DIR, "/193_054/1991/LT041930541991011001T1-SC201904020431
                     paste0(IN.DIR, "/193_055/1991/LT041930551991011001T1-SC201904020424

```

```
p193.2000 = list(paste0(IN.DIR, "/193_052/2000/LE071930522000020401T1-SC20190520100729",
                      paste0(IN.DIR, "/193_053/2000/LE071930532000020401T1-SC20190520100345",
                      paste0(IN.DIR, "/193_054/2000/LE071930542000020401T1-SC20190402045232",
                      paste0(IN.DIR, "/193_055/2000/LE071930552000020401T1-SC20190402043121"))

p193.2003 = list(paste0(IN.DIR, "/193_052/2003/LE071930522002122601T1-SC20190405182352",
                      paste0(IN.DIR, "/193_053/2003/LE071930532002122601T1-SC20190405182309",
                      paste0(IN.DIR, "/193_054/2003/LE071930542002122601T1-SC20190405182226",
                      paste0(IN.DIR, "/193_055/2003/LE071930552002122601T1-SC20190405190255"))

p193.2005 = list(paste0(IN.DIR, "/193_052/2005/LE071930522005021701T1-SC20190405190117",
                      paste0(IN.DIR, "/193_053/2005/LE071930532005021701T1-SC20190405190003",
                      paste0(IN.DIR, "/193_054/2005/LE071930542005021701T1-SC20190405182210",
                      paste0(IN.DIR, "/193_055/2005/LE071930552005021701T1-SC20190405190021"))

p193.2007 = list(paste0(IN.DIR, "/193_052/2007/LE071930522007012201T1-SC20190405182221",
                      paste0(IN.DIR, "/193_053/2007/LE071930532007012201T1-SC20190405182607",
                      paste0(IN.DIR, "/193_054/2007/LE071930542007012201T1-SC20190405182139",
                      paste0(IN.DIR, "/193_055/2007/LE071930552007012201T1-SC20190405182418"))

p193.2009 = list(paste0(IN.DIR, "/193_052/2009/LE071930522009012701T1-SC20190405182143",
                      paste0(IN.DIR, "/193_053/2009/LE071930532009012701T1-SC20190405182301",
                      paste0(IN.DIR, "/193_054/2009/LE071930542009012701T1-SC20190405182103",
                      paste0(IN.DIR, "/193_055/2009/LE071930552009012701T1-SC20190405182754"))

p193.2013 = list(paste0(IN.DIR, "/193_052/2013/LE071930522013022301T1-SC20190405182200",
                      paste0(IN.DIR, "/193_053/2013/LE071930532013022301T1-SC20190405182213",
                      paste0(IN.DIR, "/193_054/2013/LE071930542013022301T1-SC20190405182152",
                      paste0(IN.DIR, "/193_055/2013/LE071930552013022301T1-SC20190405182331"))

p193.2015 = list(paste0(IN.DIR, "/193_052/2015/LC081930522015010401T1-SC20190405181512",
                      paste0(IN.DIR, "/193_053/2015/LC081930532015010401T1-SC20190405181751",
                      paste0(IN.DIR, "/193_054/2015/LC081930542015010401T1-SC20190402042510",
                      paste0(IN.DIR, "/193_055/2015/LC081930552015010401T1-SC20190402042446"))

p193.2017 = list(paste0(IN.DIR, "/193_052/2017/LC081930522017012501T1-SC20190405181511",
                      paste0(IN.DIR, "/193_053/2017/LC081930532017012501T1-SC20190405181440",
                      paste0(IN.DIR, "/193_054/2017/LC081930542017012501T1-SC20190405181458",
                      paste0(IN.DIR, "/193_055/2017/LC081930552017012501T1-SC20190405181444"))

p193.2018 = list(paste0(IN.DIR, "/193_052/2018/LC081930522018011201T1-SC20190405181524",
                      paste0(IN.DIR, "/193_053/2018/LC081930532018011201T1-SC20190405181459",
                      paste0(IN.DIR, "/193_054/2018/LC081930542018011201T1-SC20190405181510",
                      paste0(IN.DIR, "/193_055/2018/LC081930552018011201T1-SC20190405181442"))

p193.2019 = list(paste0(IN.DIR, "/193_052/2019/LC081930522019021601T1-SC20190405183839",
                      paste0(IN.DIR, "/193_053/2019/LC081930532019021601T1-SC20190405181518",
                      paste0(IN.DIR, "/193_054/2019/LC081930542019021601T1-SC20190405183609",
                      paste0(IN.DIR, "/193_055/2019/LC081930552019021601T1-SC20190405181507"))
```

```

# Chemin WRS p194 -----
p194.1986 = list(paste0(IN.DIR, "/194_052/1986/LT051940521986011101T1-SC20190405172804
                  paste0(IN.DIR, "/194_053/1986/LT051940531986011101T1-SC20190405172758

p194.1987 = list(paste0(IN.DIR, "/194_052/1987/LT051940521986122901T1-SC20190405172903
                  paste0(IN.DIR, "/194_053/1987/LT051940531986122901T1-SC20190405174433

p194.1997 = list(paste0(IN.DIR, "/194_052/1997/LT051940521997021001T1-SC20190405181746
                  paste0(IN.DIR, "/194_053/1997/LT051940531997021001T1-SC20190405173130

p194.2000 = list(paste0(IN.DIR, "/194_052/2000/LE071940522000012601T1-SC20190405172721
                  paste0(IN.DIR, "/194_053/2000/LE071940532000012601T1-SC20190405172733

p194.2003 = list(paste0(IN.DIR, "/194_052/2003/LE071940522002121701T1-SC20190405172823
                  paste0(IN.DIR, "/194_053/2003/LE071940532002121701T1-SC20190405172739

p194.2005 = list(paste0(IN.DIR, "/194_052/2005/LE071940522004122201T1-SC20190405172700
                  paste0(IN.DIR, "/194_053/2005/LE071940532004122201T1-SC20190405172612

p194.2007 = list(paste0(IN.DIR, "/194_052/2007/LT051940522007010501T1-SC20190405172919
                  paste0(IN.DIR, "/194_053/2007/LT051940532007010501T1-SC20190405172216

p194.2010 = list(paste0(IN.DIR, "/194_052/2010/LE071940522010012101T1-SC20190405172745
                  paste0(IN.DIR, "/194_053/2010/LE071940532010012101T1-SC20190405173304

p194.2012 = list(paste0(IN.DIR, "/194_052/2012/LE071940522012011101T1-SC20190405173146
                  paste0(IN.DIR, "/194_053/2012/LE071940532012011101T1-SC20190405172236

p194.2015 = list(paste0(IN.DIR, "/194_052/2015/LC081940522015012701T1-SC20190405172055
                  paste0(IN.DIR, "/194_053/2015/LC081940532015012701T1-SC20190405172042

p194.2017 = list(paste0(IN.DIR, "/194_052/2017/LC081940522016123101T1-SC20190405172058
                  paste0(IN.DIR, "/194_053/2017/LC081940532016123101T1-SC20190405172040

p194.2018 = list(paste0(IN.DIR, "/194_052/2018/LC081940522017121801T1-SC20190405172038
                  paste0(IN.DIR, "/194_053/2018/LC081940532017121801T1-SC20190405174114

p194.2019 = list(paste0(IN.DIR, "/194_052/2019/LC081940522019012201T1-SC20190405172019
                  paste0(IN.DIR, "/194_053/2019/LC081940532019012201T1-SC20190405172055

)

# Fonction pour traiter et fusionner un ensemble d'images Landsat -----
#
# @param in.image.dirs liste des répertoires des images à traiter
# @param ext           l'étendue à utiliser pour le recadrage des images
# @param filename      nom de fichier pour la sauvegarde de l'image traitée
# @param overwrite     retraiter et écraser des images déjà existantes
# @param log           écrire les informations sur le processus dans un fichier

```

```

#
# @return          objet raster de l'image traitée (invisible)
#
prepare.landsat <- function(in.image.dirs,
                           ext=NULL,
                           filename=NULL,
                           overwrite=FALSE,
                           log=TRUE) {

  # Charger le fichier, si le fichier existe et overwrite==FALSE
  if(!is.null(filename) && file.exists(filename) && overwrite==FALSE) {
    message("- loading from file ", filename)
    out.image <- stack(filename)

  } else {

    # Ouvrir le fichier journal si un nom de fichier et log==true
    if(!is.null(filename) & log==TRUE) {
      dir.create(dirname(filename), recursive = TRUE, showWarnings = FALSE)
      logfile <- file(sub("\\\\.[[:alnum:]]+$", ".log", filename), open="wt")
      sink(logfile, type="output")
      sink(logfile, type="message")
      message(date())
    }

    # Listes vides pour les images et des bandes de qualité
    images <- list()
    qas <- list()

    # Pour chaque image ...
    for(image.dir in in.image.dirs) {

      image.sensor <- substr(basename(image.dir), 0,4)
      if(image.sensor=="LC08") {
        regexp <- "^.*_pixel_qa|band2|band3|band4|band5|band6|band7|evi|msavi|nbr|nbr2|"
      } else {
        regexp <- "^.*_pixel_qa|band1|band2|band3|band4|band5|band7|evi|msavi|nbr|nbr2|"
      }
      image <- stack(grep(regexp, dir(image.dir, full.names=TRUE), value=TRUE))
      image.name   <- substr(names(image)[1], 1, 40)
      image.scene  <- paste0(substr(image.name, 11, 13), "_", substr(image.name, 14, 16))
      image.date   <- substr(image.name, 18, 21)
      image.path   <- as.numeric(substr(image.scene, 1, 3))
      image.row    <- as.numeric(substr(image.scene, 5, 7))

      # ... renommer les couches de l'image
      names(image) <- c("qa", SST.LSBANDS)

      message("- ", image.name, ": ", appendLF = FALSE)
    }
  }
}

```



```

names(out.image) <- SST.LSBANDS
out.qa <- writeRaster(out.qa,
                      filename = sub("[.]tif$", paste0("_qa", image.sensor, ".tif"),
                      overwrite = overwrite,
                      datatype="INT2S"))
}

}

message("done")
print(out.image)

# Fermer le fichier journal
if(!is.null(filename) & log==TRUE) {
  sink(type="output")
  sink(type="message")
}

# Retourner l'image de manière invisible
invisible(out.image)

}

# COMMENCER LE TRAITEMENT #####
# Attention, peut facilement remplir le répertoire tmp !
# Peut-être seulement traiter une partie des images et ensuite redémarrer R

# Étendue du Togo + 5km de tampon en UTM 30 pour le chemin p194
TGO.EXT.30 <- extent(spTransform(TGO, UTM.30)) + 10000

# Pour chaque ensembles d'images (année/chemin), en parallèle, ...
registerDoParallel(CORES-1)
foreach(i=1:length(in.image.list)) %dopar% {

  # ... extraire la liste des images à traiter
  in.image.dirs <- in.image.list[[i]]

  name <- unlist(strsplit(names(in.image.list[i]), "[.]"))
  path <- name[1] # p.ex. "p192"
  year <- name[2] # p.ex. "1986"
  tile <- name[3] # p.ex. "NA" (ou 1,2, ...)

  # ... étendue UTM 30 pour chemin p194 et UTM 31 pour les autres
  if(path == "p194") tmp.ext <- TGO.EXT.30 else tmp.ext <- TGO.EXT

  # ... répertoire pour sauvegarder l'image
  out.image.dir <- paste0(OUT.DIR, "/", path)
  if(!dir.exists(out.image.dir)) dir.create(out.image.dir)

  # ... nom du fichier
}

```

```

if(is.na(tile)) {
  filename <- paste0(out.image.dir, "/", path, "_", year, ".tif")
} else {
  filename <- paste0(out.image.dir, "/", path, "_", year, "_", tile, ".tif")
}

# ... et faire le travail
message("Processing ", path, "_", year)
prepare.landsat(in.image.dirs,
  ext = tmp.ext,
  filename = filename,
  overwrite=FALSE,
  log=TRUE)

}

# Supprimer les fichiers temporaires dans le répertoire tmp
tmp_dir <- tempdir()
files <- list.files(tmp_dir, full.names = T, recursive=T)
file.remove(files)

# Fusionner les fichiers journaux
for(dir in dir(paste0(OUT.DIR), full.names=TRUE)) {
  path <- basename(dir)
  system(paste0("tail -n +1 ", dir, "/*.log > ", dir, "/", path, ".tmp"))
  system(paste0("rm ", dir, "/*.log"))
  system(paste0("mv ", dir, "/", path, ".tmp ", dir, "/", path, ".log"))
}
}

# Reprojection des images p194 de UTM 30 à UTM 31 =====
p194.dir <- paste0(OUT.DIR, "/p194")

# Pour chaque image p194, en parallèle, ...
registerDoParallel(CORES-1)
foreach(image=dir(p194.dir, pattern=".*[.]tif$")) %dopar% {

  image <- paste0(p194.dir, "/", image)
  image.utm30 <- sub("[.]tif$", "utm30.tif", image)
  file.rename(image, image.utm30)

# ... transformer l'image en utilisant l'outil externe "gdalwarp"
  system(paste("gdalwarp",
    image.utm30,
    "-t_srs '+proj=utm +zone=31 +datum=WGS84'",
    "-tr 30 30",
    "-te 147255 1017495 222165 1238265",
    image,
    "-ot 'Int16'",
    "-overwrite"))
}

```

```

# ... supprimer l'image UTM 30
file.remove(image.utm30)
}

# Masquer images avec bande de qualité (nuages/ombres) =====
registerDoParallel(CORES-1)
foreach(file=dir(OUT.DIR, pattern=".*\\_[:digit:]"]+\\.tif$", full.names = TRUE, recursive=TRUE)
  qa      <- raster(dir(dirname(file), pattern=gsub("\\_", "\\_", sub("\\.tif", "_qa*", basename(file))))
  image <- mask(brick(file), qa %in% c(QA.CLOUD, QA.SHADOW, QA.WATER, QA.ICE), maskvalue=0)
  writeRaster(image, sub("\\.tif", "_m.tif", file), overwrite = TRUE, datatype="INT2S",
} 

# Créer des vignettes de chaque image =====
registerDoParallel(CORES-1)
foreach(filename=dir(OUT.DIR, pattern="p19.*[.]tif$", recursive=TRUE, full.names=TRUE))
  image <- brick(filename)
  jpeg(sub("[.]tif$", ".jpeg", filename), width=1350, height=3000)
  par(plt=c(0,1,0,1))
  plot(spTransform(TGO, UTM.31))
  plotRGB(image, r=6, g=5, b=3, stretch="lin", add=TRUE)
  plot(mask(image[[1]], spTransform(TGO, UTM.31), inverse=TRUE), col="#FFFFFF66", legend=TRUE)
  plot(spTransform(TGO, UTM.31), add=TRUE, lwd=3)
  dev.off()
}

```

0.2.1.2 WorldClim

Les variables climatiques ont une grande influence sur les caractéristiques des forêts. Ils sont donc importants pour l'évaluation des surfaces forestières et de la biomasse sur la base d'images satellites. Les données climatiques de WorldClim version 2 ont servi comme base. **La température annuelle moyenne (BIO1)**, **la saisonnalité de la température (BIO4)**, **la précipitation annuelle (BIO12)** et **la saisonnalité de la précipitation (BIO15)** étant identifiées comme les variables WorldClim les plus importantes pour l'évaluation des forêts.

0.2.1.2.1 Acquisition des données

Des données climatiques historiques (1970 – 2020) avec une résolution de 30 arcsecondes (environ 1 km² sur l'équateur) disponible à WorldClim version 2 ont été utilisées.

Pour l'analyse, des **variables climatiques mensuelles** ont été utilisées :

- **prec**: Précipitations pour les mois de janvier à décembre (mm)
- **tmax**: Température maximale pour les mois de janvier à décembre (°C)
- **tavg**: Température moyenne pour les mois de janvier à décembre (°C)
- **tmin**: Température minimale pour les mois de janvier à décembre (°C)

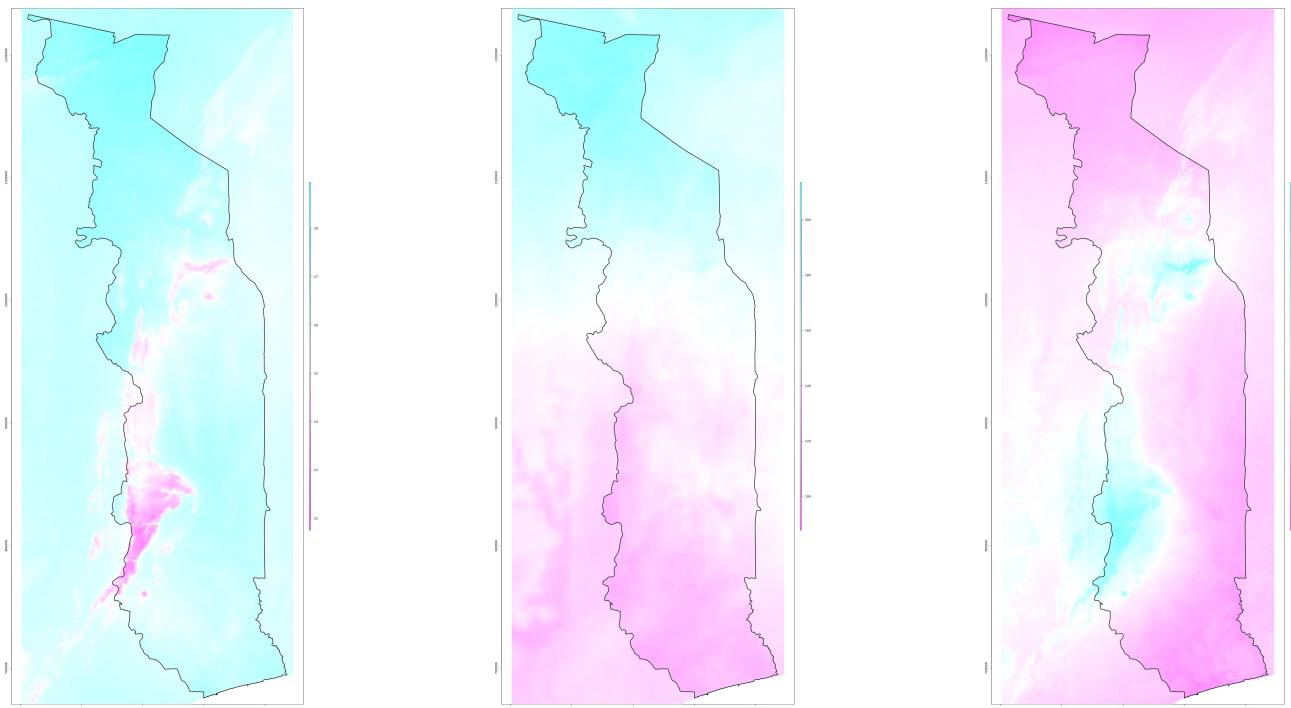
Et également les **variables bioclimatiques** qui en découlent :

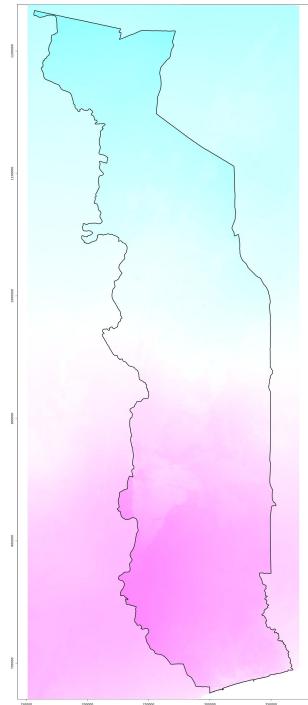
- **BIO1:** Température moyenne annuelle
- **BIO2:** Fourchette diurne moyenne (moyenne des températures mensuelles (max temp - min temp))
- **BIO3:** Isothermie ($\text{BIO2}/\text{BIO7}$) ($\times 100$)
- **BIO4:** Saisonnalité de la température (écart-type $\times 100$)
- **BIO5:** Température maximale du mois le plus chaud
- **BIO6:** Température minimale du mois le plus froid
- **BIO7:** Gamme de température annuelle ($\text{BIO5}-\text{BIO6}$)
- **BIO8:** Température moyenne du trimestre le plus humide
- **BIO9:** Température moyenne du trimestre le plus sec
- **BIO10:** Température moyenne du trimestre le plus chaud
- **BIO11:** Température moyenne du trimestre le plus froid
- **BIO12:** Précipitation annuelle
- **BIO13:** Précipitation du mois le plus humide
- **BIO14:** Précipitation du mois le plus sec
- **BIO15:** Saisonnalité de la précipitation (Coefficient de variation)
- **BIO16:** Précipitation du trimestre le plus humide
- **BIO17:** Précipitation du trimestre le plus sec
- **BIO18:** Précipitation du trimestre le plus chaud
- **BIO19:** Précipitation du trimestre le plus froid

0.2.1.2.2 Prétraitement des données

Les données sont lues et reprojetées sur le raster des images Landsat (UTM 31, résolution de 30 mètres) et coupées à la taille.

0.2.1.2.2.1 Example





Données bioclimatiques WorldClim version 2: température annuelle moyenne (BIO1) / saisonnalité de la température (BIO4) / précipitation annuelle (BIO12) / saisonnalité de la précipitation (BIO15)

0.2.1.2.3 Script R: 01_SSTS/01_data/_src/prep-Worldclim.R

```
#####
# prep-Worldclim.R: lire et reprojeter les données de WorldClim
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables =====

IN.DIR  <- paste0(DIR.RAW.DAT, "/Worldclim")
OUT.DIR <- paste0(DIR.SST.DAT, "/Worldclim")
if(!dir.exists(OUT.DIR)) dir.create(OUT.DIR)

# Reprojection images WorldClim vers Landsat (résolution 30m, UTM 31, ...) ===

foreach(file=dir(IN.DIR, pattern=".*Togo[.]tif$")) %dopar% {
  system(paste("gdalwarp",
    paste0(IN.DIR, "/", file),
    "-t_srs '+proj=utm +zone=31 +datum=WGS84'",
    "-tr 30 30",
    paste("-te", TGO.EXT@xmin, TGO.EXT@ymin, TGO.EXT@xmax, TGO.EXT@ymax),
    paste0(OUT.DIR, "/", file),
    "-dstnodata -3.4e+38",
```

```

    "-co COMPRESS='LZW',
    "-co INTERLEAVE='BAND',
    "-overwrite"))

system(paste("gdalinfo -stats",
             paste0(OUT.DIR, "/", file)))
}

# Créer des vignettes pour les données WorldClim =====

foreach(file=dir(OUT.DIR, pattern=".*[.]tif$")) %dopar% {
  image <- stack(paste0(OUT.DIR, "/", file))
  type <- unlist(strsplit(file, "_"))[3]
  if      (type == "prec") { zlim <- c(0,320);   col <- rev(topo.colors(255)) }
  else if (type == "tmin") { zlim <- c(14.0,27.8); col <- rev(heat.colors(255)) }
  else if (type == "tmax") { zlim <- c(24.9,37.5); col <- rev(heat.colors(255)) }
  else if (type == "tavg") { zlim <- c(19.7,32.7); col <- rev(heat.colors(255)) }
  else                      { zlim <- NA;           col <- rev(cm.colors(255)) }
  foreach(i=1:nlayers(image)) %dopar% {
    jpeg(paste0(OUT.DIR, "/", sub("[.]tif$", "", file), "-",
                str_pad(i, 2, "left", 0),
                width=1350, height=3000)
    plot(image[[i]], col=col, zlim=zlim)
    plot(mask(image[[i]], TGO, inverse=TRUE), col="#FFFFFF66", legend=FALSE, add=TRUE)
    plot(TGO, add=TRUE, lwd=3)
    dev.off()
  }
}
}

```

0.2.1.3 SRTM

Les variables topographiques telles que l'**élévation, la pente et l'aspect** ne sont actuellement utilisées ni pour l'évaluation des forestières ni pour l'évaluation de la biomasse, car il a été constaté que ces variables ont peu d'influence sur le pouvoir explicatif des modèles de classification ou de régression. Néanmoins, les données sont conservées et disponibles pour autres analyses.

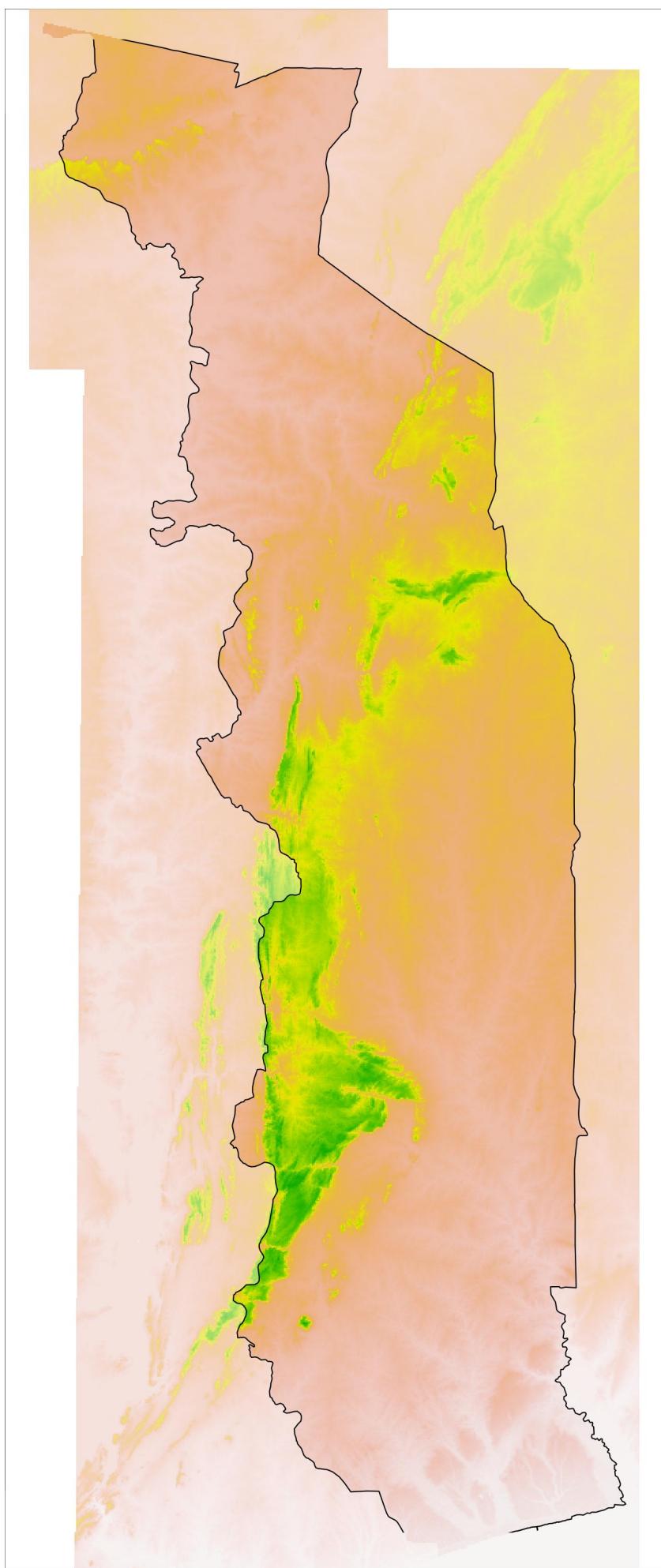
0.2.1.3.1 Acquisition des données

Les données topographiques du SRTM ont été obtenues à partir de deux sources:

- **Les données originales avec une résolution spatiale de 1 seconde d'arc** (environ 30 mètres sur l'équateur) ont été obtenues du jeu de données SRTM 1 Arc-Second Global disponible sur USGS Earthexplorer. Ces données à haute résolution ont des fois des lacunes (manque des données).
- **Les données SRTM ajustées avec une résolution de 3 secondes d'arc** ont été obtenues à partir de CGIAR SRTM 90m Digital Elevation Database v4.1 pour combler les lacunes des données à haute résolution.

0.2.1.3.2 Prétraitement des données

Les données de 1 seconde d'arc sont lues et les lacunes éventuelles sont comblées avec les données ajustées de 3 secondes d'arc. Enfin, les données sont reprojetées sur le raster des images Landsat (UTM 31, résolution de 30 mètres) et coupées à la taille.



0.2.1.3.3 Script R: 01_SSTS/01_data/_src/prep-SRTM.R

```
#####
# prep-SRTM.R: lire, nettoyer et empiler des données topographiques SRTM
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables =====
IN.DIR <- paste0(DIR.RAW.DAT, "/SRTM")
OUT.DIR <- paste0(DIR.SST.DAT, "/SRTM")
if(!dir.exists(OUT.DIR)) dir.create(OUT.DIR)

# Préparation du modèle numérique d'élévation SRTM =====

# Lire 90m SRTM MNE sans vides (source: http://srtm.csi.cgiar.org/srtmdata/),
dem.90 <- do.call(merge, lapply(as.list(dir(paste0(IN.DIR, "/3arcsecond")),
                                         pattern=".*[.]tif$",
                                         full.names=TRUE)), raster))

# Lire 30m SRTM MNE (source: USGS Earthexplorer) et remplir vides avec 90m SRTM
dem.30 <- foreach(tile=dir(paste0(IN.DIR, "/1arcsecond")), pattern=".*[.]tif$", full.names=TRUE,
                     .combine=merge, .multicombine=TRUE) %dopar% {
  dem.30.t <- raster(tile)
  dem.90.t <- round(projectRaster(dem.90, dem.30.t))
  merge(dem.30.t, dem.90.t)
}

# Reprojection d'images MNE vers Landsat (résolution 30m, UTM 31, ...)
writeRaster(dem.30, paste0(OUT.DIR, "/SRTM-1arcsec_raw.tif"),
            datatype="INT2S",
            overwrite=TRUE)
system(paste("gdalwarp",
            paste0(OUT.DIR, "/SRTM-1arcsec_raw.tif"),
            "-t_srs '+proj=utm +zone=31 +datum=WGS84'",
            "-tr 30 30",
            paste("-te", TGO.EXT@xmin, TGO.EXT@ymin, TGO.EXT@xmax, TGO.EXT@ymax),
            paste0(OUT.DIR, "/SRTM-1arcsec.tif"),
            "-ot 'Int16'",
            "-co COMPRESS='LZW'",
            "-co INTERLEAVE='BAND'",
            "-overwrite"))
file.remove(paste0(OUT.DIR, "/SRTM-1arcsec_raw.tif"))

# Calculer les statistiques GDAL (min, max, ...)
```

```

system(paste("gdalinfo -stats", paste0(OUT.DIR, "/SRTM-1arcsec.tif")))

# Créer une vignettes du MNE =====
dem <- raster(paste0(OUT.DIR, "/SRTM-1arcsec.tif"))
jpeg(paste0(OUT.DIR, "/SRTM-1arcsec.jpeg"), width=1350, height=3000)
par(plt=c(0,1,0,1))
plot(dem)
plot(mask(dem, TGO, inverse=TRUE), col="#FFFFFF66", legend=FALSE, add=TRUE)
plot(TGO, add=TRUE, lwd=3)
dev.off()

```

0.2.1.4 SoilGrids

Les données SoilGrids sur le type de sol et le carbone du sol n'ont pas encore été incluses dans les analyses de la surface forestière et des modifications du stockage du carbone dues au changement d'utilisation des terres. Ils sont disponibles pour des analyses complémentaires.

0.2.1.4.1 Acquisition des données

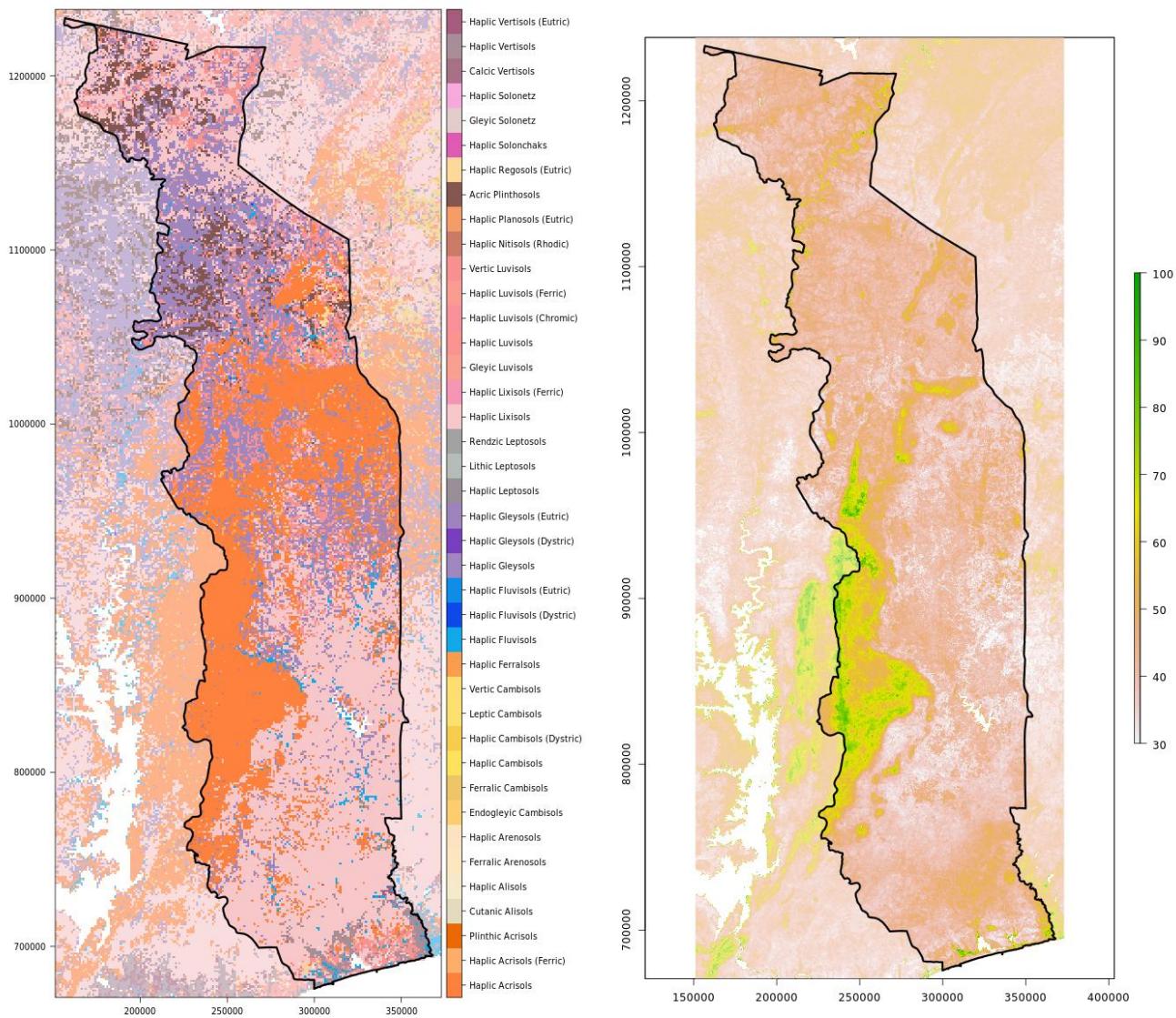
Les données sur les sols proviennent des archives SoilGrids (documentation SoilGrids 2017)

0.2.1.4.2 Prétraitement des données

Les données sont lues et reprojetées sur le raster des images Landsat (UTM 31, résolution de 30 mètres) et coupées à la taille.

0.2.1.4.2.1 Example

Données pédologiques SoilGrids: Typologie des sols WRB à gauche, Carbone du sol à 30 cm de profondeur (tC/ha) à droite



0.2.1.4.3 Script R: 01_SSTS/01_data/_src/prep-SoilGrids.R

```
#####
# prep-SoilGrids.R: lire et reprojeter des données SoilGrids
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Source des données: https://files.isric.org/soilgrids/data/recent/
# Documentation: https://www.isric.org/explore/soilgrids/faq-soilgrids-2017

# Définitions des variables =====

IN.DIR <- paste0(DIR.RAW.DAT, "/SoilGrids")
OUT.DIR <- paste0(DIR.SST.DAT, "/SoilGrids")
if(!dir.exists(OUT.DIR)) dir.create(OUT.DIR)
```

```

# Reprojection oilGrids vers Landsat (résolution 30m, UTM 31, ...) =====

foreach(file=dir(IN.DIR, pattern=".*[.]tif$")) %do% {
  system(paste("gdalwarp",
    paste0(IN.DIR, "/", file),
    "-t_srs '+proj=utm +zone=31 +datum=WGS84'",
    "-tr 30 30",
    paste("-te", TGO.EXT@xmin, TGO.EXT@ymin, TGO.EXT@xmax, TGO.EXT@ymax),
    paste0(OUT.DIR, "/", file),
    "-dstnodata -3.4e+38",
    "-co COMPRESS='LZW'",
    "-co INTERLEAVE='BAND'",
    "-overwrite"))

  system(paste("gdalinfo -stats",
    paste0(OUT.DIR, "/", file)))
}

# Créer des vignettes pour les données SoilGrids =====

# Types de sol WRB -----
jpeg(paste0(OUT.DIR, "/TAXNWRB_250m_ll.jpeg"), width=675, height=1200)
soil.types <- raster(paste0(OUT.DIR, "/TAXNWRB_250m_ll.tif"))
soil.types <- as.factor(soil.types)
cats <- levels(soil.types)[[1]]
attr <- read.csv(paste0(IN.DIR, "/TAXNWRB_250m_ll.tif.csv"))
attr <- merge(cats, attr[,c("Number", "Group", "WRB_group", "R", "G", "B")],
  by.x = "ID", by.y = "Number")
cats[["Soil Type"]] <- attr$Group
levels(soil.types) <- cats
levelplot(soil.types,
  col.regions=paste0("#", as.hexmode(tmp$R),as.hexmode(tmp$G),as.hexmode(tmp$B)),
  xlab="", ylab="") +
  levelplot(mask(soil.types, TGO, inverse=TRUE), col.regions="#FFFFFF66") +
  layer(sp.polygons(TGO, lwd=3))
dev.off()

# Réservoir carbone à 30 cm -----
jpeg(paste0(OUT.DIR, "/OCSTHA_M_30cm_250m_ll.jpeg"), width=675, height=1200)
image <- raster(paste0(OUT.DIR, "/OCSTHA_M_30cm_250m_ll.tif"))
raster::plot(image, zlim=c(30,100))
raster::plot(mask(image, TGO, inverse=TRUE), col="#FFFFFF66", legend=FALSE, add=TRUE)
sp::plot(TGO, add=TRUE, lwd=3)
dev.off()

# Tableau des surfaces des catégories de sol selon GIEC =====
tmp <- as.data.frame(table(mask(soil.types, TGO)[] * 30^2/10000)

```

```

tmp <- merge(tmp, attr, by.x="Var1", by.y="ID")
tmp$IPCC <- NA
tmp$IPCC[tmp$WRB_group %in% c("Leptosols", "Vertisols", "Kastanozems", "Chernozems",
                                "Phaeozems", "Luvisols", "Alisols", "Albeluvisols",
                                "Solonetz", "Calcisols", "Gypsisols", "Umbrisols",
                                "Cambisols", "Regosols")] <- "HAC"
tmp$IPCC[tmp$WRB_group %in% c("Acrisols", "Lixisols", "Nitisosols", "Ferralsols",
                                "Durisols")] <- "LAC"
tmp$IPCC[tmp$WRB_group %in% c("Arenosols")] <- "SAN"
tmp$IPCC[tmp$WRB_group %in% c("Podzols")] <- "POD"
tmp$IPCC[tmp$WRB_group %in% c("Andosols")] <- "VOL"
tmp$IPCC[tmp$WRB_group %in% c("Gleysols")] <- "WET"
tmp$IPCC[is.na(tmp$WRB_group)] <- "Other"

```

0.2.1.5 ProREDD

Dans le cadre du premier inventaire forestier national de 2015/16, le projet GIZ “ProREDD” a réalisé une **carte d’occupation des sols à partir d’images RapidEye des années 2013/14** (Rapport méthodes et résultats).

La carte n’est pas utilisée pour l’analyse de l’évolution des surfaces forestières dans le cadre du NRF-MRV REDD+, mais elle est disponible comme carte comparative et pour d’autres analyses.

0.2.1.5.1 *Acquisition des données*

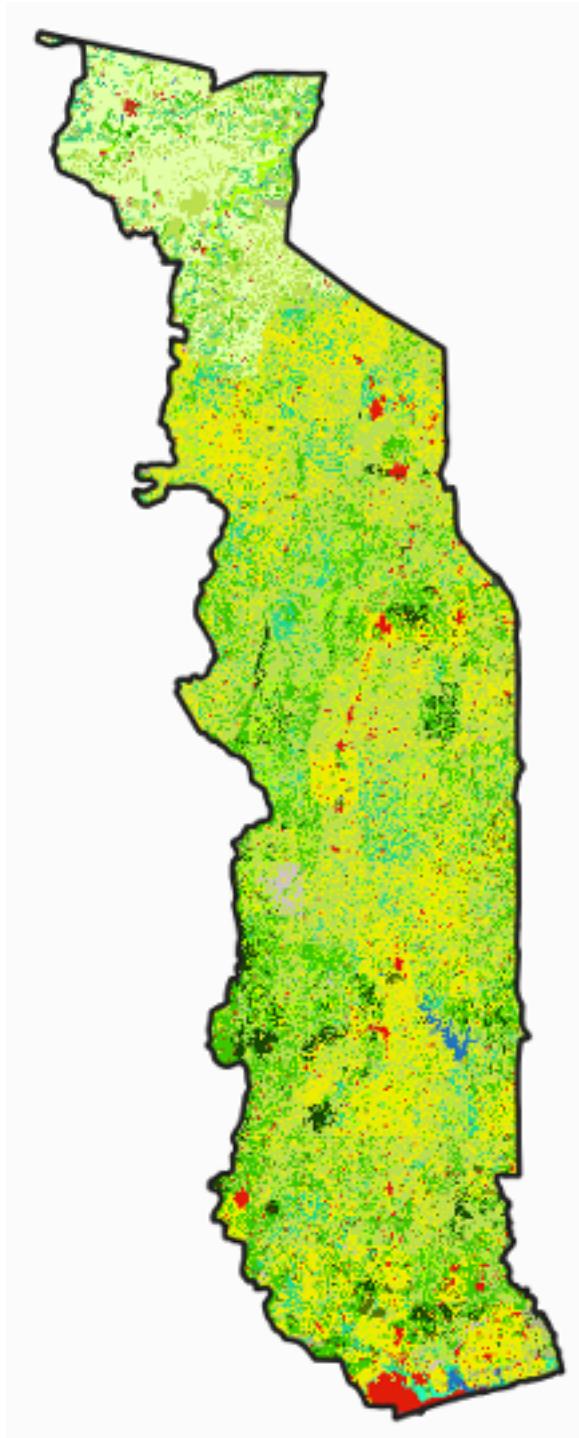
Les cartes pour les différentes régions ont été fournies par L’Unité de gestion de bases de données cartographiques (UGBDC) de la Direction des études et de la planification (DEP) le ministère sous forme de Shapefiles.

0.2.1.5.2 *Prétraitement des données*

Les shapefiles sont lus et convertis avec l’outil “grid_gridding” de SAGA GIS en données raster avec résolution originale des images RapidEye de 5 mètres, puis reprojetés sur le raster Landsat de 30 mètres.

0.2.1.5.2.1 *Example*

xl



0.2.1.5.3 Script R: 01_SSTS/01_data/_src/prep-ProREDD.R

```
#####
# prep-ProREDD.R: rasterizer la carte d'occupation des terres RapidEye
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020
```

```

# Définitions des variables =====

IN.DIR <- paste0(DIR.RAW.DAT, "/RapidEye/shapefiles")
OUT.DIR <- paste0(DIR.SST.DAT, "/ProREDD")
if(!dir.exists(OUT.DIR)) dir.create(OUT.DIR)

# Rasterizer Shapefiles avec l'outil grid_gridding de SAGA =====

files <- dir(IN.DIR, pattern=".shp$", recursive=TRUE, full.names=TRUE)
registerDoParallel(CORES - 1)
foreach(file=files) %dopar%{
  out.file <- sub("shp$", "sdat", file)
  system(paste0("saga_cmd grid_gridding \"Shapes to Grid\" ",
    "-TARGET_DEFINITION 0 -INPUT \"", file, "\" ",
    "-FIELD \"code\" -OUTPUT 2 -MULTIPLE 0 -LINE_TYPE 0 ",
    "-POLY_TYPE 0 -GRID_TYPE 2 -TARGET_USER_SIZE 30.0 ",
    "-TARGET_USER_FITS 0 -GRID \"", out.file, "\""))
}

# Lire et fusionner les différentes scènes raster
scenes <- lapply(dir(IN.DIR, pattern=".sdat$", recursive=TRUE, full.names=TRUE), raste
scenes["tolerance"] <- 0.4
RE <- do.call(merge, scenes)

# À réviser: Reprojection sur l'image Landsat
AGB <- raster("../output/3_forest-biomass/2_ref-maps/TGO_2015_AGB.tif")
RE_resampled <- resample(RE, AGB, method="ngb")
writeRaster(RE_resampled, paste0(OUT.DIR, "/TGO_30m.tif"), datatype="INT2S")

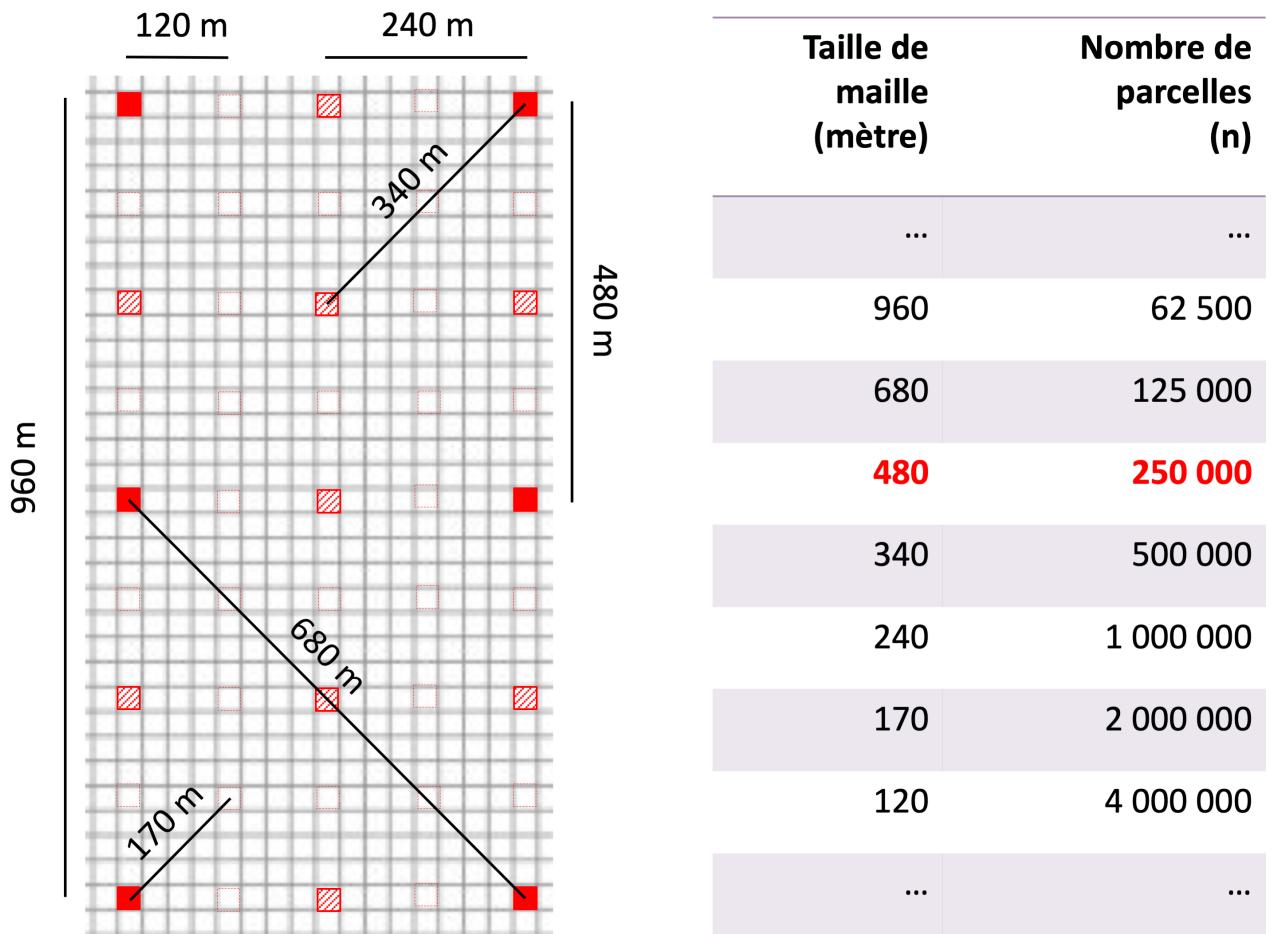
```

0.2.2 Collecte des données

0.2.2.1 Réseau d'échantillonnage

Les parcelles d'échantillonage couvrent une surface de 30 x 30 mètres et **correspondent en taille, forme et position aux pixels des images Landsat**. Les parcelles d'échantillonnage sont régulièrement réparties sur le territoire du Togo, sur une grille d'une taille de maille de 480 mètres (tous les 16 pixels Landsat) ou 4,3 parcelles d'échantillonnage par km². Sur l'ensemble du territoire togolais, cela donne un réseau de 250 000 parcelles.

Si nécessaire pour autres utilisation, la taille de maille de 480 mètres permet d'enlargir la maille à 679 ou 960 mètres. D'autre part, il est également possible de comprimer la maille à 340, 240, 170, 120, 85, ... mètres.



La grille d'échantillonnage est basée sur l'étendue du Togo (alignée avec la grille Landsat) et la taille du maillage. Les points d'échantillonnage résultants sont attribués avec les coordonnées x et y et l'ID correspondant. La grille de points résultante est enregistrée sous forme de shapefile (télécharger archive ZIP).

0.2.2.1.1 Script R: 01_SSTS/02_BdD/_src/create-grid.R

```
#####
# create-grid.R: créer une grille de points d'observation SSTS
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables =====

OUT.DIR <- paste0(DIR.SST.BDD, "/01_reseau-SSTS")
if(!dir.exists(OUT.DIR)) dir.create(OUT.DIR)

RES <- 480 # Résolution de la grille (mètres)

# Grille d'observation =====
```

```

# Coordonnées min/max en ligne avec images Landsat
x.min <- RES * extent(TGO)@xmin %/% RES
x.max <- RES * extent(TGO)@xmax %/% RES
y.min <- RES * extent(TGO)@ymin %/% RES
y.max <- RES * extent(TGO)@ymax %/% RES

# Creation de la grille
frame.points <- SpatialPoints(expand.grid(seq(x.min, x.max, by=RES),
                                             seq(y.min, y.max, by=RES)),
                                 proj4string=UTM.31) [TGO]

# Ajouter attribues PLOTID, xcoords et ycoords
frame.points$xcoords <- frame.points@coords[,1]
frame.points$ycoords <- frame.points@coords[,2]
frame.points$PLOTID <- paste0(str_pad(frame.points@xcoords, 7, "left", "0"), "_",
                               str_pad(frame.points@ycoords, 7, "left", "0"))

# Sauvegarder comme fichier Shapefile
writeOGR(frame.points, dsn=OUT.DIR, layer="TGO_frame_480m",
          driver="ESRI Shapefile", overwrite=TRUE)

```

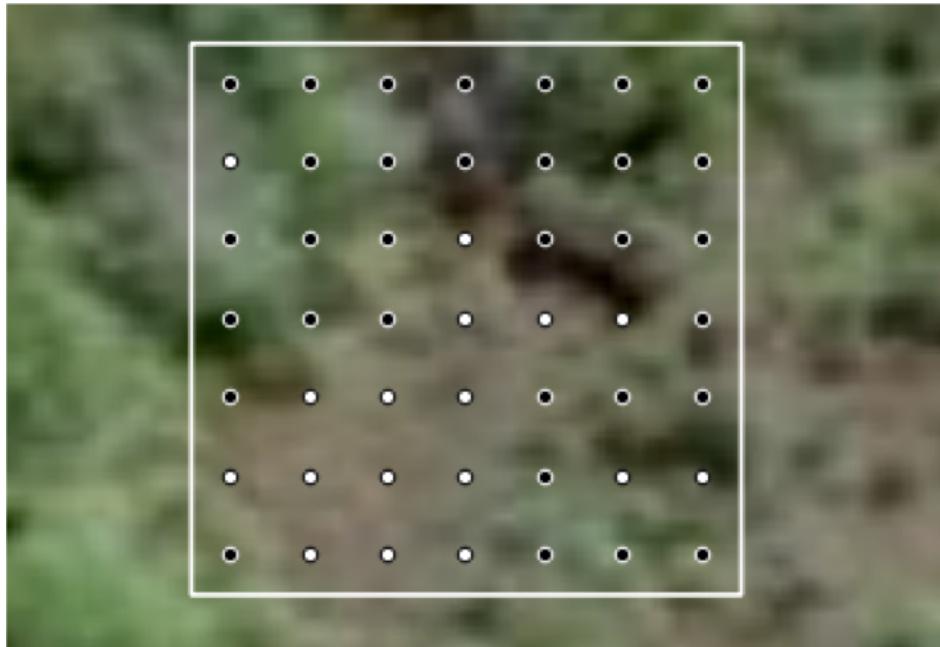
0.2.2.2 Parcelles d'entraînement

0.2.2.2.1 Échantillonnage

Les parcelles d'entraînement sont sélectionnées de manière à couvrir la plus large gamme possible de couverture végétale sur l'ensemble du territoire du Togo. À cette fin, le **NDVI comme indicateur du couvert végétal** est attribué à tout les points du réseau d'échantillonnage et stratifiées en 10 classes NDVI. Un échantillon aléatoire spatialement équilibré de 1 500 échantillons est ensuite tiré de chaque strate NDVI (generalized random tessellation).

0.2.2.2.2 Couverture des houppiers

Pour déterminer la couverture des houppiers une **grille de 7 x 7 points** est définie à l'intérieur de chaque parcelles d'échantillonage. Pour chaque point de ce grille, les photo-interprètes déterminent sur base des images de très haute résolution disponible en GoogleEarth, si le **point touche l'houppier d'un arbre ou non**. La couverture des houppiers est en suite déterminé par le nombre des points qui tombent sur un arbre par rapport au nombre total des points ($n = 49$).



L’attribution est fait par les photo-interprètes en QGIS, avec l’image GoogleEarth comme carte de base. La source et la date d’acquisition de l’image GoogleEarth utilisé est également enregistré. Elle est obtenu par GoogleEarth Pro (plugin QGIS `send2google_earth`).

0.2.2.2.3 Script R: `01_SSTS/02_BdD/_src/create-train-plots.R`

```
#####
# create-train-plots.R: Créer un ensemble de parcelles d'entraînement
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Préparation des variables =====

OUT.DIR <- paste0(DIR.SST.BDD, "/02_train-plots/empty")
if(!dir.exists(OUT.DIR)) dir.create(OUT.DIR, recursive=TRUE)

# Charger la grille d'échantillonnage du SSTS
frame.points <- readOGR(paste0(DIR.SST.BDD, "/01_reseau-SSTS/TGO_frame_480m.shp"))

# Charger NDVI 2018 masquée (sans nuages, ombre, ...)
ndvi.band <- which(SST.LSBANDS == "ndvi")
ndvi <- merge(raster(paste0(DIR.SST.DAT, "/Landsat/p192/p192_2018_m.tif")), band=ndvi.band,
              raster(paste0(DIR.SST.DAT, "/Landsat/p193/p193_2018_m.tif")), band=ndvi.band,
              raster(paste0(DIR.SST.DAT, "/Landsat/p194/p194_2018_m.tif")), band=ndvi.band)

# Extraire le NDVI pour chaque parcelle et le découper en 10 strates
frame.points$ndvi    <- raster::extract(ndvi, frame.points)
frame.points$ndvi_c <- cut(frame.points$ndvi, 10, labels=paste0("s", 0:9))
```

```

# Echantillonnage des parcelles d'entraînement =====

# Initialiser le générateur de nombres aléatoires
set.seed(RSEED)

# Définition d'échantillonnage (1500 échantillons par strate NDVI s0 - s9)
Dsgn.grt <- list("s0"=list(panel=c(PanelOne=1500), seltype="Equal"),
                  "s1"=list(panel=c(PanelOne=1500), seltype="Equal"),
                  "s2"=list(panel=c(PanelOne=1500), seltype="Equal"),
                  "s3"=list(panel=c(PanelOne=1500), seltype="Equal"),
                  "s4"=list(panel=c(PanelOne=1500), seltype="Equal"),
                  "s5"=list(panel=c(PanelOne=1500), seltype="Equal"),
                  "s6"=list(panel=c(PanelOne=1500), seltype="Equal"),
                  "s7"=list(panel=c(PanelOne=1500), seltype="Equal"),
                  "s8"=list(panel=c(PanelOne=1500), seltype="Equal"),
                  "s9"=list(panel=c(PanelOne=1500), seltype="Equal"))
)

# Échantillonnage spatialement équilibrée (Generalized Random Tessellation)
train.points <- grts(sp.object=frame.points, # grille d'échantillonnage
                      src.frame="sp.object", # type de grille (objet spatial)
                      design=Dsgn.grt, # définition d'échantillonnage
                      stratum="ndvi_c", # colonne avec strate
                      type.frame="finite", # type d'échantillonnage
                      DesignID="train" # préfixe pour chaque point
)

# Appliquer le système de référence des coordonnées
proj4string(train.points) <- proj4string(frame.points)

# Mélanger les points d'entraînement et ajouter un identifiant
train.points <- train.points[sample(1:nrow(train.points)), ]
train.points$SAMPLEID <- paste0("trn-", str_pad(string=1:nrow(train.points),
                                                width = 4,
                                                pad = "0",
                                                side = "left"))

# Conversion des points en parcelles et ajouter des attributs =====

# Grille Landsat
landsat.grid <- raster(ndvi)
values(landsat.grid) <- 1

# Selectionner les pixels Landsat correspondantes et convertir en polygone
train.plots <- rasterToPolygons(mask(landsat.grid, train.points))

# Extraire les attributs des points d'entraînement ...
train.plots@data <- over(train.plots,

```

```

            train.points[, c("PLOTID", "SAMPLEID",
                            "xcoords", "ycoords",
                            "ndvi", "stratum")])

# ... et compléter avec autres attributs
train.plots$ccov <- as.character(NA) # couverture houppiers
train.plots$img_src <- train.plots$img_date <- as.character(NA) # source et date d'image
train.plots$author <- train.plots$mod_date <- as.character(NA) # auteur et date

# Sauvegarder parcelles sous format Shapefile et KML
writeOGR(train.plots, dsn=OUT.DIR, layer="COV_parcelles", driver="ESRI Shapefile", overwir...
writeKML(train.plots, kmlname="COV_parcelles", filename=paste0(OUT.DIR, "/COV_parcelles.kml"))

# Créer une grille d'échantillon 7x7 dans chaque parcelle =====
# Déterminer la grille
grid.size <- 7
res <- res(landsat.grid)[1]
offset <- c(res/grid.size/2 + (0:(grid.size-1))*res/grid.size)

# Diviser les parcelles pour un traitement parallèle
subsets <- split(train.plots, f=1:(CORES-1))
registerDoParallel(CORES-1)
train.grids <- foreach(subset=subsets, .combine=rbind, .multicombine=TRUE) %dopar% {
  # Créer un couche de points vides ...
  grids <- SpatialPointsDataFrame(data.frame(x = 0, y = 0),
                                    data=data.frame(PLOTID = 0,
                                                    SAMPLEID = 0,
                                                    GRIDPOINT = 0))[-1,]

  # ... et ajoute la grille d'échantillon pour chaque parcelle
  for(p in 1:length(subset)) {
    plot <- subset[p,]
    ext <- extent(plot)
    grids <- bind(grids, SpatialPointsDataFrame(expand.grid(ext@xmin+offset, ext@ymin+o...
                                                data=data.frame(PLOTID = plot$PLOTID,
                                                                SAMPLEID = plot$SAMPLEID,
                                                                GRIDPOINT = 1:grid.size))
  }
  grids
}

# Appliquer le système de référence des coordonnées
proj4string(train.grids) <- proj4string(train.plots)

# Ajouter un attribut "arbre" (1: oui, 0: non)
train.grids$tree <- as.integer(NA)

# Sauvegarder les grille d'échantillon sous format Shapefile
writeOGR(train.grids, dsn=OUT.DIR, layer="COV_parcelles_grid",

```

```

driver="ESRI Shapefile", overwrite=TRUE)

# Diviser parcelles et grilles d'échantillon en 10 sous-ensembles =====
# pour le traitement par différents photo-interprètes

subsets <- split(train.plots, f=1:10)
for(i in 1:length(subsets)) {
  # Sauvegarder parcelles sous format Shapefile et KML
  writeOGR(subsets[[i]], dsn=OUT.DIR, layer=paste0("COV_parcelles_", i),
            driver="ESRI Shapefile", overwrite=TRUE)
  writeKML(subsets[[i]], kmlname=paste0("COV_parcelles_", i) ,
            filename=paste0(OUT.DIR, "/COV_parcelles_", i, ".kml"))
  # Sauvegarder les grilles correspondantes sous format Shapefile
  subset.grids <- train.grids[train.grids$PLOTID %in% subsets[[i]]$PLOTID,]
  writeOGR(subset.grids, dsn=OUT.DIR, layer=paste0("COV_parcelles_", i, "_grid"),
            driver="ESRI Shapefile", overwrite=TRUE)
}

```

0.2.2.3 Parcelles de validation

0.2.2.3.1 Échantillonnage

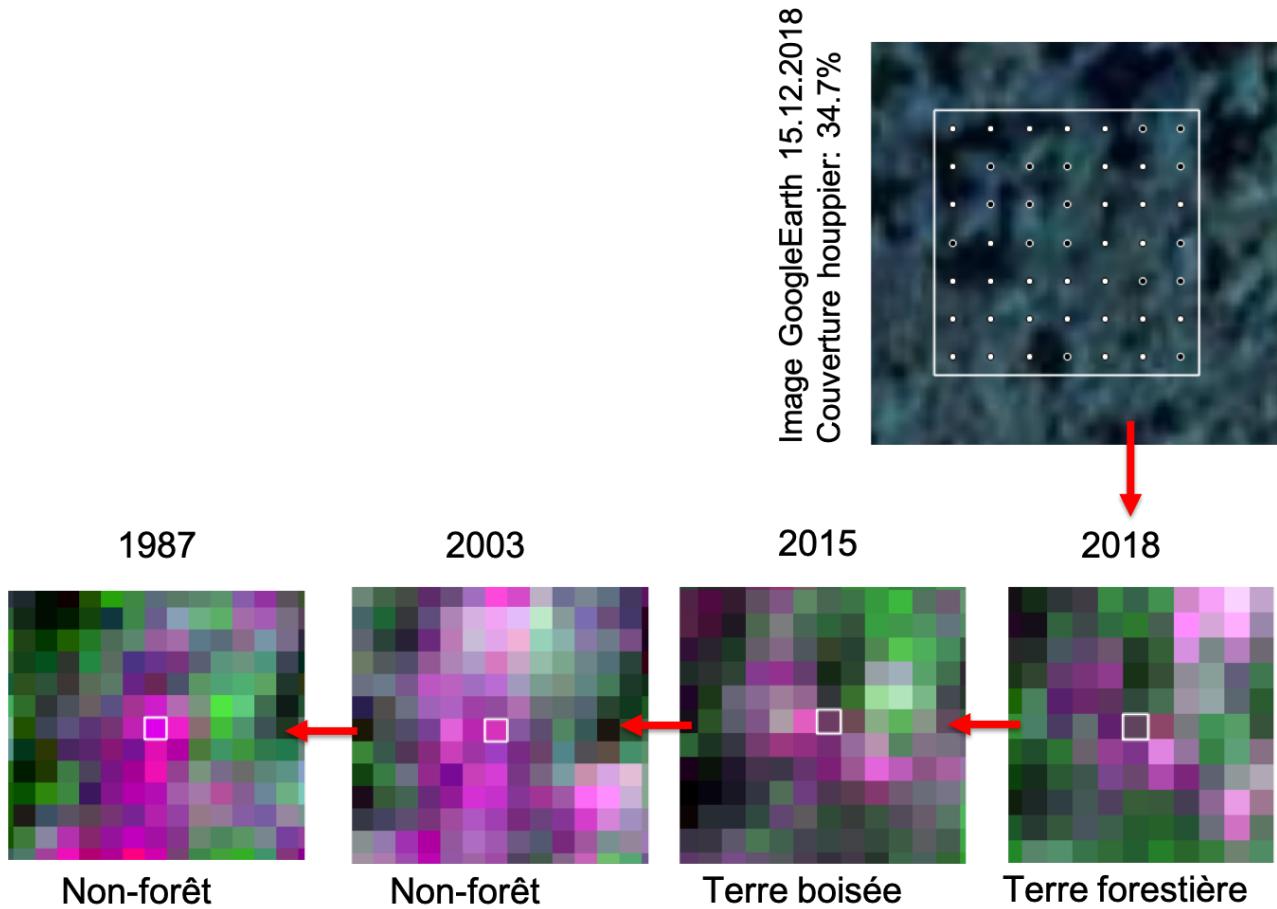
La sélection des parcelles de validation est basée sur les cartes forêt/non-forêt, notamment leurs transitions entre les années 1987 – 2003 – 2015 – 2018. Tout d'abord les transitions sont déterminées pour chaque point du réseau d'échantillonnage. Ensuite un échantillon aléatoire stratifié est tirré avec une répartition de l'échantillon aux strates (transitions) qui est la valeur moyenne d'une répartition proportionnelle à la taille des strates et d'une répartition égale.

0.2.2.3.2 Occupation des terres

L'occupation des terres et le changement de l'occupation des terres est **déterminé sur base des images Landsat**. Trois différentes catégories sont distinguées:

- forêt (couverture houppier $\geq 30\%$)
- terre boisée (couverture houppier entre 10% et 30%)
- non-forêt (couverture houppier $< 10\%$)

La couverture des houppiers est utilisé pour déterminer l'occupation des terres sur l'image Landsat de la date correspondante. À partir de cette référence, l'occupation des terres est déterminé pour les autres dates de référence 1987 – 2003 – 2015 – 2018. La figure suivante illustre la procédure:



0.2.2.3.3 Script R: 01_SSTS/02_BdD/_src/create-val-plots.R

```
#####
# create-val-points.R: Créer un ensemble de parcelles de validation
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Préparation des variables =====

OUT.DIR <- paste0(DIR.SST.BDD, "/03_val-plots/empty")
if(!dir.exists(OUT.DIR)) dir.create(OUT.DIR, recursive=TRUE)

IN.DIR <- paste0(DIR.MRV.MCF, "/2_raw-maps/FC30/TGO")

# Fusionner les cartes des années de référence (1987, 2003, 2015 et 2018)
maps <- merge(raster(paste0(IN.DIR, "/TGO_1987_F30r.tif")),
               raster(paste0(IN.DIR, "/TGO_2003_F30r.tif")),
               raster(paste0(IN.DIR, "/TGO_2015_F30r.tif")),
               raster(paste0(IN.DIR, "/TGO_2018_F30r.tif")))
```

```

# Créer carte des changements (p.ex. FFFN pour déforestation entre 2015 et 2018)
change.map <- maps[[1]] + maps[[2]]*10 + maps[[3]]*100 + maps[[4]]*1000

# Charger la grille d'échantillonnage du SSTS
frame.points <- readOGR(paste0(DIR.SST.BDD, "/01_reseau-SSTS/TGO_frame_480m.shp"))

# Extraire la transition pour chaque parcelle
frame.points$trans <- raster::extract(change.map, frame.points)

# Allocation des points de validation =====
n <- 4000                                # la taille de l'échantillon
alloc <- freq(change.map)[1:16,]          # fréquence des transitions (p. ex FFFN)

# Allocation proportionnelle et égale
alloc <- cbind(alloc,
               prop=round(n*alloc[, "count"] / sum(alloc[, "count"])),
               equal=round(n/nrow(alloc)))

# Allocation balancée (moyenne entre proportionnelle et égale)
alloc <- cbind(alloc,
               balanced=round((alloc[, "prop"] + alloc[, "equal"]) / 2))

# Echantillonnage des parcelles de validation =====
# Ajoute les attributs des parcelles d'entraînement
train.plots      <- readOGR(paste0(DIR.SST.BDD, "/02_train-plots/assessed/COV_parcell"))
frame.points     <- merge(frame.points,
                           train.plots@data[,c("PLOTID", "img_date", "img_src", "mod_d",
                           by="PLOTID", all.x=TRUE)

# Crée une couche de points vide
val.points <- frame.points[0,]

# Initialiser le générateur de nombres aléatoires
set.seed(RSEED)

# Pour chaque transition ...
for(i in 1:nrow(alloc)) {

  strat.n <- alloc[i, "balanced"]    # nombre d'échantillons à tirer
  sample.ids <- NULL                 # vecteur pour les ID à échantillonner

  # Tout d'abord, prend les parcelles avec ...
  ids.tp <- which(!is.na(frame.points$trans)
                  & frame.points$trans==alloc[i, "value"]
                  & !is.na(frame.points$ccov))    # couverture houppier connue
  n.tp   <- min(length(ids.tp), strat.n)
}

```

```

if (n.tp > 0) sample.ids <- c(sample.ids, sample(ids.tp, n.tp))

# ... et compléter avec autres échantillons de la grille avec
ids.r <- which(!is.na(frame.points$trans) &
               frame.points$trans==alloc[i, "value"] &
               is.na(frame.points$ccov))      # couverture houppier inconnue
n.r    <- min(length(ids.r), strat.n - n.tp)
if (n.r > 0) sample.ids <- c(sample.ids, sample(ids.r, n.r)) # aléatoirement

# Ajouter aux points de validation
val.points <- rbind(val.points, frame.points[sample.ids, ])
}

# Mélanger les points de validation et ajouter un identifiant
val.points <- val.points[sample(1:nrow(val.points)), ]
val.points$SAMPLEID <- paste0("val-", str_pad(string=1:nrow(val.points),
                                              width = 4,
                                              pad = "0",
                                              side = "left"))

# Conversion des points en parcelles et ajouter des attributs =====
# Grille Landsat
landsat.grid <- raster(change.map)
values(landsat.grid) <- 1

# Selectionner les pixels Landsat correspondantes et convertir en polygone
val.plots <- rasterToPolygons(mask(landsat.grid, val.points))

# Extraire les attributs des points d'entraînement ...
val.plots@data <- over(val.plots, val.points[, c("PLOTID", "SAMPLEID",
                                                 "xcoords", "ycoords",
                                                 "trans", "ccov",
                                                 "img_src", "img_date",
                                                 "author", "mod_date")])

# ... convertir couverture des houppiers disponibles en %, les autres NA
val.plots$ccov <- format(round(100*val.plots$ccov, 1))
val.plots$ccov[val.plots$ccov == " NA"]<- NA

# ... et compléter avec attributs occupation des terre 1987, 2003, 2015 et 2018
val.plots$lc_18 <- val.plots$lc_15 <- as.character(NA)
val.plots$lc_03 <- val.plots$lc_87 <- as.character(NA)

# Sauvegarder parcelles sous format Shapefile et KML
writeOGR(val.plots, dsn=". ", layer="UOT_parcelles", driver="ESRI Shapefile", overwrite=TRUE)
writeKML(val.plots, kmlname="UOT_parcelles", filename="UOT_parcelles.kml")

```

```

# Créer une grille d'échantillon 7x7 dans chaque parcelle =====

# Déterminer la grille
grid.size <- 7
res <- res(landsat.grid)[1]
offset <- c(res/grid.size/2 + (0:(grid.size-1))*res/grid.size)

# Diviser les parcelles pour un traitement parallèle
subsets <- split(val.plots, f=1:86)
registerDoParallel(CORES-1)
val.grids <- foreach(subset=subsets, .combine=bind, .multicombine=TRUE) %dopar% {
  # Créer un couche de points vides ...
  grids <- SpatialPointsDataFrame(data.frame(x = 0, y = 0),
                                   data=data.frame(PLOTID = 0,
                                                   SAMPLEID = 0,
                                                   GRIDPOINT = 0))[-1,]
  # ... et ajoute la grille d'échantillon pour chaque parcelle
  for(p in 1:length(subset)) {
    plot <- subset[p,]
    ext <- extent(plot)
    grids <- bind(grids, SpatialPointsDataFrame(expand.grid(ext@xmin+offset, ext@ymin+
                                                               data=data.frame(PLOTID = plot$PLOTID,
                                                               SAMPLEID = plot$SAMPLEID,
                                                               GRIDPOINT = 1:grid.size^
                                                               }
    grids
  }
}

# Appliquer le système de référence des coordonnées
proj4string(val.grids) <- proj4string(val.plots)

# fusionner avec les attributs (arbre, oui ou non?) déjà collectés
train.grids <- readOGR(paste0(DIR.SST.BDD, "/02_train-plots/assessed/COV_parcelles_grid"))
val.grids <- merge(val.grids,
                    train.grids@data[, c("PLOTID", "GRIDPOINT", "tree")],
                    all.x=TRUE)

# Sauvegarder les grille d'échantillon sous format Shapefile
writeOGR(val.grids, dsn=OUT.DIR, layer="UOT_parcelles_grid", driver="ESRI Shapefile", ov

# Diviser parcelles et grilles d'échantillon en 10 sous-ensembles =====
# pour le traitement par différents photo-interprètes

subsets <- split(val.plots, f=1:10)
for(i in 1:length(subsets)) {
  # Sauvegarder parcelles sous format Shapefile et KML
  writeOGR(subsets[[i]], dsn=OUT.DIR, layer=paste0("UOT_parcelles_", i),
            driver="ESRI Shapefile", overwrite=TRUE)
  writeKML(subsets[[i]], kmlname=paste0("UOT_parcelles_", i) ,
}

```

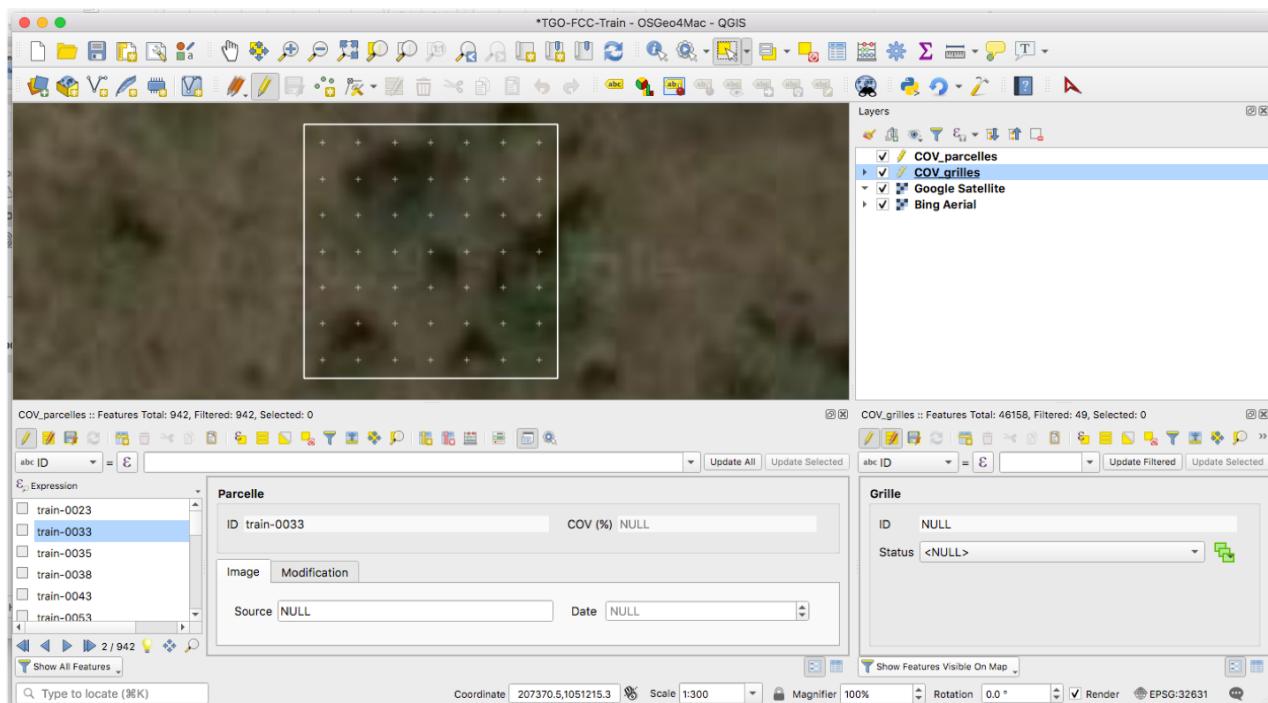
```

filename=paste0("UOT_parcelles_", i, ".kml"))
# Sauvegarder les grilles correspondantes sous format Shapefile
subset.grids <- val.grids[val.grids$PLOTID %in% subsets[[i]]$PLOTID,]
writeOGR(subset.grids, dsn=OUT.DIR, layer=paste0("UOT_parcelles_", i, "_grid"),
          driver="ESRI Shapefile", overwrite=TRUE)
}

```

0.2.2.4 Outil QGIS

Actuellement, les parcelles d'échantillonnage du SSTS ainsi que la grille de points pour déterminer la couverture des houppiers sont stockées sous forme des fichiers Shapefile. L'acquisition des attributs est effectuée par les photo-interprètes à l'aide d'un formulaire défini dans un projet QGIS.



Un SSTS basé sur des fichiers limite la complexité de la structure des données (par exemple, l'enregistrement récurrent des mêmes parcelles sur des images de différentes années) ainsi que le travail parallèle de différents photo-interprètes sur les mêmes données. Actuellement, des travaux sont en cours pour transférer le SSTS vers une base de données géographiques (PostGIS) sur un serveur central au Ministère de l'Environnement et des Ressources Forestières (MERF) et pour définir un formulaire QGIS qui permet à plusieurs personnes de travailler simultanément sur les données.

0.3 Inventaire Forestier National

L'inventaire forestier national, combine les données collectées directement sur le terrain avec des références forestières provenant de différentes sources. Le seul ensemble de données collectées est actuellement le premier IFN-1 national inverntaire forestier qui a été réalisé par le GIZ en 2015/16. Il est utilisé dans le cadre du NRF-MRV pour déterminer les facteurs d'émission de la biomasse des arbres.



Un deuxième inverntaire forestier national (IFN-2) sur base des parcelles permanentes installées par IFN-1 est en cours de planification. Ce deuxième inventaire sera realisé par la direction de l'environnment du Mininstère de l'environnement, du développement durable et de la protection de la nature.

En outre, d'autres ensembles de données seront intégrés dans la base de données au cours des prochaines années:

- inventaires réalisés par les universités du Togo et autres acteurs
- inventaire des plantations de l'ODEF
- données sur les feux de végétation de l'ANGE
- données sur l'agriculture (superficie emblavées et le cheptel) de DCID et ITRA
- données démographiques de INSEED
- ...

Les données et les documents des enquêtes respectives sont structurés comme suit:

```

02_IFN          # INVENTAIRE FORESTIER NATIONAL =====
|--- 01_IFN-1   # Premier IFN 2015/16 -----
    |--- 01_data  # données brutes
    |--- 02_aux   # données auxiliaires, scripts, analyses
    |--- 03_reports # rapports (méthode, résultats, ...)
    |_-- ...
    |_-- ...
  
```

0.3.1 IFN-1 (2015/16)

Le premier inventaire forestier national a ete réalisé par la GIZ et le MERF en 2015/16.

0.3.1.1 Méthode

L'inventaire IFN-1 a installé xx parcelles permanentes au Togo. Les parcelles ont été installée ...

0.3.1.2 Données

0.3.1.3 Résultats

Les résultats ...

0.4 Analyses NRF/MRV

Le Togo a soumis son premier **Niveau de Référence pour les Forêts (NRF v1.0)** en Janvier 2020 (*Soumissions du Togo sur la plateforme web REDD+ de la CCNUCC*). Il est basé sur l'évolution de la couverture forestier entre 2003 – 2018 (pertes et gains des terres forestières à une couverture du houppier $\geq 30\%$) et le stockage de carbone dans la biomasse aérienne des arbres (incl. bois mort sur pied)

Les données de base qu'on a utilisé pour effectuer ce travail sont:

- les **images Landsat de l'archive USGS (1985 – 2019)** et les **données climatiques WorldClim version 2** pour la la cartographie de l'évolution des surfaces forestiers et la cartographie de la biomasse
- les **données du SSTS (état 2019)** sur la couverture du houppier et l'utilisation des terres pour la calibration et la validation des cartes sur l'évolution des surfaces forestier
- les **données dendrométriques du IFN-1 (2015/16)** pour déterminer le stockage de carbone dans la biomasse aérienne par parcelle et la calibration des cartes de la biomasse

Ce chapitre décrit **l'approche méthodologique et les outils techniques** utilisés pour établir ce NRF. C'est un travail en cours. Le même approche sera utilisés a) pour améliorer le NRF avec des nouvelles données et/ou méthodes et b) pour mettre à jour régulièrement les analyses dans le cadre du Monitoring, reporting et vérification (MRV).

Chaque section commence par une description de la méthodologie utilisée et des références aux points clés dans le script R correspondant, suivie par une présentation exemplaire des résultats de cette étape et enfin du code R commenté.

- **Section 0.2.1** décrit l'acquisition et la préparation des données de télédétection dans le cadre du Système de Surveillance Terrestre par Satellite SSTS (Images Landsat, données WorldClim, ...).
- **Section 0.2.2** décrit la collecte de données d'entraînement et de validation dans le cadre du SSTS.
- **Section 0.4.1** décrit les différentes étapes nécessaires pour la cartographie des surfaces forestières et leur évolution: la classification des séries de cartes, leur nettoyage et validation.
- **Section 0.4.2** décrit les différentes étapes nécessaires pour la cartographie des la biomasse aérienne: l'évaluation des données de l'IFN, la calibration des cartes de biomasse leur nettoyage et l'analyse de l'évolution de la biomasse dans le temps.

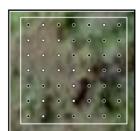
0.4.1 Analyse surfaces forestiers

L'analyse des surfaces forestiers est fait par une **classification supervisée**. Les données du SSTS sur la couverture des houppiers (section 0.2.2.2) est utilisé pour calibrer un modèle de classification *RandomForest* sur base des images satellitaires Landsat et les données climatiques Worldclim v2. Tous les parcelles d'entraînement avec une **couverture des houppiers $\geq 30\%$** sont considérées comme forêt, les autres comme non-forêts.

Le schéma suivant montre les étapes pour la production des cartes forêt/non-forêt sur toute la période 1986 – 2019:

Parcelles d'entraînement

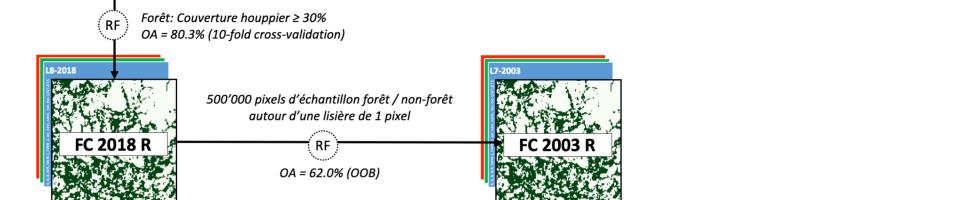
Couverture houppier sur des pixels Landsat (30 x 30m) en 2017 – 2018 sur base des images GoogleEarth (n = 7'488, stratifié selon NDVI)

**Classification de l'évolution du couvert forestier**

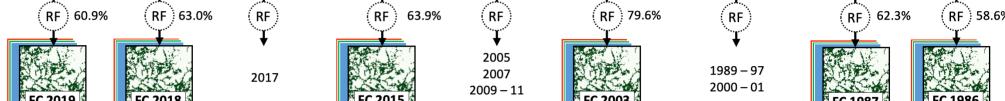
Sur la base des images Landsat 1986 – 2019

Cartes de référence

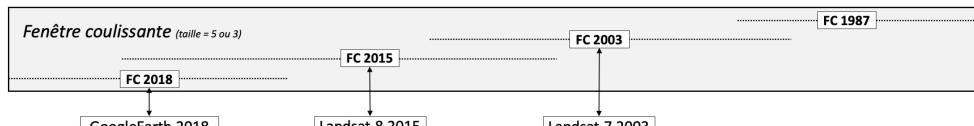
carte forêt / non-forêts 2018 sur base des parcelles d'entraînement
carte carte forêt / non-forêts 2003 calibré avec carte 2018

**Classification multi-date**

Niveau 1:
Série des cartes brutes

**Niveau 2:**

Nettoyage temporelle (lissage et filtrage jachères)

**Validation**

Pixels de validation (n = 2'413, stratifié selon classes de transition)

GoogleEarth 2018 (OA = 89.0%)
Landsat-8 2015 (OA = 87.9%)
Landsat-7 2003 (OA = 86.6%)

Vue que les données sur la couverture des houppier et seulement disponible pour les années récentes (à cause d'une disponibilité limité des images de très haute résolution), c'est seulement la carte forêt/non-forêt 2018 qui a été produit sur base des parcelles d'entraînement du SSTS. Cette **carte de référence 2018 est utilisé comme base pour la calibration des modèles de classification pour les autres dates** pour lesquelles des images Landsat sont disponible.

Tout d'abord la carte de référence 2018 est utilisé pour calibrer la classification d'une carte forêt/non-forêt en 2003. C'est cette **carte de référence 2003 qu'on a utilisé pour calibrer les autres cartes de 1986 – 2018**. On a choisi de faire la classification de toute la série des images sur la carte de référence 2003, parce que les images Landsat 2003 sont de très bonne qualité (et donc la carte est probablement aussi de bonne qualité) et parce que l'année 2003 se trouve au milieu de la période analysée. Si on prend directement la carte forêt/non-forêt 2018 comme référence pour la calibration de la série 1986 – 2018, on observe une généralisation de la surface forestier le plus on s'éloigne de la date 2018, donc un changement de la surface forestier qui est plutôt un artefact de la méthode que une changement d'occupation des terres.

Dans la suite, la **série des cartes forêts/non-forêts 1986 – 2018 est nettoyé par une lyssage temporelle**, pixel par pixel, avec un filtre majoritaire (fenêtre coulissante d'une taille de 5). Finalement, une reforestation est seulement constaté si on a observé la forêt pour une période de 10 ans et plus.

Les **cartes nettoyés de 2003, 2015 et 2018 sont validés avec les données SSTS** sur l'occupation des terres (Section 0.2.2.3.2) et les matrices d'erreurs sont utilisées pour déterminer la précision des cartes individuelles et des différents changements d'occupation des terres, en utilisant la méthode de Olofsson et al. (2014)

0.4.1.1 Production des cartes Forêt/Non-Forêt

La **production de la carte forêt/non-forêt 2018** est basée sur la couverture des houppiers déterminée dans les parcelles d'entraînement, les images satellitaires Landsat (bandes et indices B, G, R, NIR, SWIR1, SWIR2, nbr, ndmi, ndvi, evi) et les données climatiques Worldclim (BI01, BI04, BI012, BI015).

Dans une première étape, les variables Landsat et Worldclim sont extraites pour toutes les **par-**

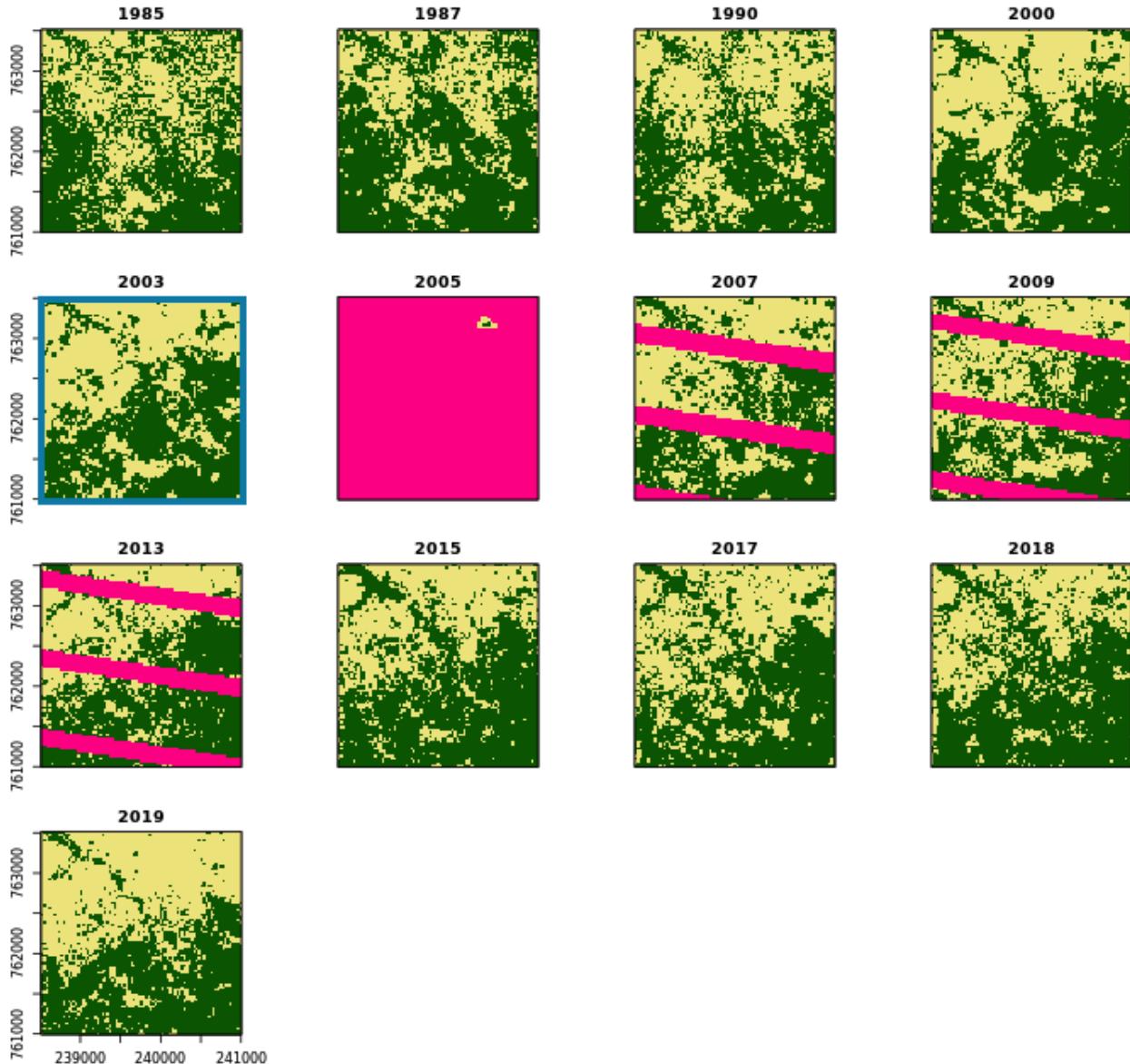
celles d'entraînement de la période de référence 1.1.2017 - 31.12.2018 et les parcelles d'entraînement sont classées comme “forestières” ou “non forestières” en fonction de leur couverture des houppiers (forêt $\geq 30\%$ de couverture des houppiers). Par la suite, des cartes forêt/non-forêt 2018 sont produites pour chaque chemin WRS. Pour cela, la fonction `classify.image()` est utilisée, qui crée un modèle de classification en **utilisant l'algorithme RandomForest** et une carte correspondante. Pour le chemin WRS central p193, l'algorithme de classification est calibré uniquement sur la base des parcelles d'entraînement. Pour les chemins p192 et p194, des données d'entraînement supplémentaires basées sur la carte p193 sont utilisées pour assurer la calibration entre les chemins.

Les cartes de référence générées pour 2018 sont maintenant utilisées pour **générer les cartes correspondantes pour l'année 2003. Les données d'entraînement sont tirées de la carte de référence 2018, autour de la lisière des forêts.** Pour les chemins p192 et p194, de nouvelles données d'entraînement supplémentaires du chemin p193 sont utilisées.

Sur la base des cartes forêt/non-forêt de 2003, les cartes de toutes les années avec des images Landsat disponibles sont produites selon la même procédure. Le calibrage de toute la série sur la base d'une année de référence garantit une différenciation consistante entre les classes forêt et non-forêt. Enfin, les chemins p192, p193 et p194 sont fusionnés pour les années clés 1987, 2003, 2015 et 2018.

0.4.1.1.1 Example

La figure suivante illustre la **série des 13 cartes forêt/non-forêt brutes** dans une région au sud de Kpalimé. Les pixels en rose vif sont des pixels dont les données sont manquantes (nuages, ombres, L7 SLC-off). Voir série des cartes nettoyées pour comparaison.



0.4.1.1.2 Script R: 03_NRF-MRV/01_MCF/_src/01_create-FC-maps.R

```
#####
# 01_create-fc-maps.R: créer des cartes brutes du couvert forestier
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables =====

# Seuil de la couverture des houppiers (forêt vs. non-forêt)
COV.FC      <- 30

# Nombre de pixels à considerer comme lisière forêt
SAMPLE.DIST <- 1
```

```

# Nombre de pixels non-NA (sera déterminé plus tard)
N.PIXELS      <- NA

# Part de pixels à prendre en compte pour la calibration des cartes
SAMPLE.RATIO   <- 0.0025

# Part des pixels à prendre des autres chemins (calibration)
CAL.RATIO      <- 0.75

# Bandes à utiliser pour la modélisation forêt vs. non-forêt
PREDICTORS    <- c("B", "G", "R", "NIR", "SWIR1", "SWIR2",
                  "nbr", "ndmi", "ndvi", "evi",
                  "BIO1", "BIO4", "BIO12", "BIO15")

# Répertoires
LANDSAT.DIR <- DIR.SST.DAT.LST
WORLDCLIM.DIR <- DIR.SST.DAT.WC2
REF.DIR <- DIR.MRV.MCF.REF
RAW.DIR <- DIR.MRV.MCF.RAW

# Définitions des fonctions =====
# Charger un image Landsat -----
# 
# @param filename Chemin du fichier landsat
# 
# @return          Image Landsat avec bandes nommées
# 

load.image <- function(filename) {
  image <- brick(paste0(LANDSAT.DIR, filename))
  names(image) <- SST.LSBANDS
  return(image)
}

# Tirer des points d'entraînement d'une carte autour de la lisière forêt -----
# 
# @param map        Carte forêt/non-forêt à échantillonner
# @param n          Nombre d'échantillons à tirer
# 
# @return          Points d'échantillon avec attribut forêt/non-forêt
# 

sample.map <- function(map, n) {

  tmp.src  <- tempfile(pattern = "", fileext = ".tif")      # tmp carte forêt/non-forêt
  tmp.dst1 <- tempfile(pattern = "", fileext = ".tif")      # tmp lisière à l'extérieur
  tmp.dst3 <- tempfile(pattern = "", fileext = ".tif")      # tmp lisière à l'intérieur
}

```

```

# Écrire la carte sur le disque
map <- writeRaster(map, tmp.src)

# Forêt + lisière à l'extérieur de la forêt
system(paste("gdal_proximity.py", tmp.src, tmp.dst1,
             "-values", FOREST,
             "-use_input_nodata YES",
             "-maxdist ", SAMPLE.DIST,
             "-fixed-buf-val", NONFOR))
dst1 <- raster(tmp.dst1)
NAvalue(dst1) <- 65535
cat("    ")

# Non-forêt + lisière à l'intérieur de la forêt
system(paste("gdal_proximity.py", tmp.src, tmp.dst3,
             "-values", NONFOR,
             "-use_input_nodata YES",
             "-maxdist ", SAMPLE.DIST,
             "-fixed-buf-val", FOREST))
dst3 <- raster(tmp.dst3)
NAvalue(dst3) <- 65535

# Masquer la carte avec les lisières
map <- mask(map, dst1)
map <- mask(map, dst3)

# Supprimer les fichiers temporaires
unlink(c(tmp.src, tmp.dst1, tmp.dst3))

# Echantillonnage stratifié des lisières forêt et non-forêt
n.classes <- length(unique(map))
cat(paste0(" -Sampling map (n=", n.classes, "*", round(n/n.classes), ") ... "))
sample.pts <- sampleStratified(map, round(n/n.classes), sp=TRUE)[,-1]
names(sample.pts) <- "CLASS"
cat("done\n")
return(sample.pts)
}

# Classification d'une image -----
#
# @param image      Image Landsat à classifier
# @param filename   Nom de fichier pour la sauvegarde de la carte
# @param bioclim    Raster des variables bioclimatiques à utiliser
# @param train.pts  Points d'entraînement
# @param ref.map    Carte de référence
# @param n.ref.map  Nombre de points à échantillonner
# @param cal.map    Carte de calibration (autre chemin WRS)
# @param n.cal.map  Nombre de points à échantillonner
# @param mask        Masque à utiliser pour ref.map et cal.map

```

```

# @param preds      Variables à utiliser pour le modèle
# @param type       Modèle de classification ou de regression
# @param crossval   Faire validation croisée (3 * 10-fold)
# @param prob       Produire également carte de probabilité
# @param n.cores    Nombre de processeurs à utiliser pour prédiction
#
# @return           List avec les éléments
#                   - model,
#                   - carte forêt/non-forêt
#                   - carte des probabilités (si disponible)
#



classify.image <- function(image, filename, bioclim=NULL, train pts=NULL,
                           ref.map=NULL, n.ref.map=NULL,
                           cal.map=NULL, n.cal.map=NULL,
                           mask=NULL, preds=NULL, type="classification",
                           crossval=FALSE, prob=FALSE, n.cores=8) {

  # Ouvrir le fichier journal
  txtfile <- paste0(sub("[.]tif$", "", filename), ".txt")
  cat("-- Image classification: ", basename(filename), "/",
      date(), " --\n", file=txtfile)

  # Charger des points d'entraînement -----
  if(!is.null(train.pts)) {
    cat("  -Loading training points ... ")
    train.pts <- train.pts[,1]      # utiliser que la première colonne ...
    names(train.pts) <- "CLASS"     # ... et nommer "CLASS"
    set_ReplCRS_warn(FALSE)
    proj4string(train.pts) <- proj4string(image)  # Système de coordonnées CRS
    cat("done\n")
    cat("Training points:", nrow(train.pts), "\n", file=txtfile, append=TRUE)
  }

  # Ajouter des points d'une carte de référence -----
  if(!is.null(ref.map)) {
    cat(paste0("  -Masking / buffering reference map ... \n"))
    # ... couper/masquer avec l'image
    ref.map <- mask(crop(ref.map, image[[1]]), crop(image[[1]], ref.map))
    # ... et masque additionnelle (si disponible)
    if(!is.null(mask)) ref.map <- mask(ref.map, mask)
    # Découper la carte de calibration (si disponible)
    if(!is.null(cal.map)) {
      tmp <- extend(crop(cal.map, ref.map), ref.map)
      ref.map <- mask(ref.map, tmp, inverse=TRUE)
    }
    cat("  ")
    # Tirer des points d'échantillon ...
    ref.pts <- sample.map(ref.map, n.ref.map)
    cat("Ref-map points: ", nrow(ref.pts), "/", ref.map@file@name, "/",
        "
```

```

    SAMPLE.DIST, "px\n", file=txtfile, append=TRUE)
# ... et ajouter aux points d'entraînement
if(is.null(train.pts)) {
  train.pts <- ref.pts
} else {
  train.pts <- rbind(train.pts, ref.pts)
}
}

# Ajouter des points d'une carte de calibration -----
if(!is.null(cal.map)) {
  cat(paste0(" -Masking / buffering calibration map ... \n"))
  # ... couper/masquer avec l'image
  cal.map <- mask(crop(cal.map, image[[1]]), crop(image[[1]], cal.map))
  # ... et masque additionnelle (si disponible)
  if(!is.null(mask)) cal.map <- mask(cal.map, mask)
  cat(" ")
  # Tirer des points d'échantillon ...
  cal.pts <- sample.map(cal.map, n.cal.map)
  cat("Cal-map points: ", nrow(cal.pts), "from", cal.map@file@name, "/",
      SAMPLE.DIST, "px\n", file=txtfile, append=TRUE)
  # ... et ajouter aux points d'entraînement
  if(is.null(train.pts)) {
    train.pts <- cal.pts
  } else {
    train.pts <- rbind(train.pts, cal.pts)
  }
}
# Nombre total des points d'entraînement
cat("Total points: ", nrow(train.pts), "\n", file=txtfile, append=TRUE)

# Extraire les variables correspondantes -----
# Utiliser toutes les variables si non-spécifiées dans les paramètres
if(is.null(preds)) {
  preds <- names(image)
  if(!is.null(bioclim)) preds <- c(preds, names(bioclim))
}
# Extraire les variables Landsat ...
cat(" -Extracting pixel values for bands:", preds, "... ")
train.pts <- raster::extract(image, train.pts, sp=TRUE)
# ... et Bioclim
if(!is.null(bioclim)) train.pts <- raster::extract(bioclim, train.pts, sp=TRUE)
# Ignorer des lignes avec NAs
train.dat <- na.omit(train.pts@data)[, c("CLASS", preds)]

# Calibration du modèle Random Forest -----
# Variable catégorielle -> mode de classification, autrement -> régression
if(type=="classification") train.dat[,1] <- as.factor(train.dat[,1])
cat("done\n")
cat(" -Calibrating RandomForest ... ")

```

```

sink(txtfile, append=TRUE)
# Utiliser caret::train pour validation croisée (a besoin de beaucoup de temps)
if(crossval) {
  map.model.cv <- train(y = train.dat[,1],
                         x = train.dat[,-1],
                         method = "rf",                      # RandomForest
                         importance = TRUE,
                         trControl = trainControl(
                           method = "repeatedcv",
                           number = 10,                      # k-fold
                           repeats = 3))                     # répétitions

  print(map.model.cv)
  map.model <- map.model.cv$finalModel
  print(map.model)
  cat("\n")
  print(varImp(map.model, scale=FALSE))
# autrement randomForest directement
} else {
  map.model <- randomForest(y=train.dat[,1], x=train.dat[,-1], importance=TRUE) # , do
  #
  # Parallélisation de RandomForest : cpossible, mais confusion, err.rate, mse et rs
  # https://stackoverflow.com/questions/14106010/parallel-execution-of-random-forest
  # map.model <- foreach(ntree=rep(100, 5), .combine=randomForest::combine, .multicor
  #                   randomForest(x=ref.pts[,!(names(ref.pts) == "CLASS")], y=ref.pts
  #
  print(map.model)
  cat("\n")
  print(varImp(map.model))
}

sink()
# Mesures des erreurs
if(type=="treecover") {
  cat("R2:", round(map.model$rsq[500], 2), "RMSE:", round(sqrt(map.model$mse[500]), 2)
} else {
  cat("OOB error rate:", round(map.model$err.rate[500,1], 2), "\n")
}

# Classification de la carte forêt/non-forêt -----
dir.create(dirname(filename), recursive=TRUE, showWarnings=FALSE)
cat("    -Creating map ... ")
# empiler les couches Landsat et bioclim
if(!is.null(bioclim)) image <- stack(image, crop(bioclim, image))
# classifier l'image sur différents processeurs en parallèle
beginCluster(n=n.cores)
map <- clusterR(image, predict, args=list(model=map.model))
endCluster()
# sauvegarder la carte
cat("writing map ... ")
# convertir en %, si c'est une carte couverture houppier
if(type=="treecover") map <- floor(map*100)

```

```

map <- writeRaster(map, filename=filename, format="GTiff", datatype="INT2U", overwrite=TRUE)
cat("done\n")

# Calculer une carte de probabilité -----
if(prob==TRUE) {
  cat("  -Creating probability map ... ")
  beginCluster(n=n.cores)
  prob.map <- clusterR(image, predict, args=list(model=map.model, type="prob"))
  endCluster()
  cat("writing map ... ")
  writeRaster(prob.map, filename=sub("\\.tif", "_prob.tif", filename), format="GTiff",
  cat("done\n"))
} else {
  prob.map <- NULL
}

cat("-- Done: ", basename(filename), "/", date(), " --\n", file=txtfile, append=TRUE)

invisible(list(
  "model"  = map.model,
  "map"    = map,
  "prob"   = prob.map
))
}

# COMMENCER LE TRAITEMENT #####
# Charger images 2018 -----
ref.p192 <- brick(paste0(LANDSAT.DIR, "/p192/p192_2018_m.tif"))
ref.p193 <- brick(paste0(LANDSAT.DIR, "/p193/p193_2018_m.tif"))
ref.p194 <- brick(paste0(LANDSAT.DIR, "/p194/p194_2018_m.tif"))
names(ref.p192) <- names(ref.p193) <- names(ref.p194) <- SST.LSBANDS
ref.images <- list(p192=ref.p192, p193=ref.p193, p194=ref.p194)

# Déterminer le nombre de pixels non-NA
N.PIXELS <- list(p192 = ncell(ref.p192[["B"]]) - summary(ref.p192)[["NA's","B"]],
                  p193 = ncell(ref.p193[["B"]]) - summary(ref.p193)[["NA's","B"]],
                  p194 = ncell(ref.p194[["B"]]) - summary(ref.p194)[["NA's","B"]])

# Charger variables bioclim
bioclim.p192 <- brick(paste0(WORLDCLIM.DIR, "/p192/p192_bioclim.tif"))
bioclim.p193 <- brick(paste0(WORLDCLIM.DIR, "/p193/p193_bioclim.tif"))
bioclim.p194 <- brick(paste0(WORLDCLIM.DIR, "/p194/p194_bioclim.tif"))
names(bioclim.p192) <- names(bioclim.p193) <- names(bioclim.p194) <- SST.BIOCLIM
bioclim <- list(p192=bioclim.p192, p193=bioclim.p193, p194=bioclim.p194)

# Charger points d'entraînement -----
train.plots <- readOGR(paste0(DIR.SST.BDD.TPS, "/COV_parcelles.shp"))

```

```

train.plots <- train.plots[!is.na(train.plots$ccov), # couverture des houppiers!
                           c("PLOTID", "ccov", "img_date", "author")]

# Convertir les polygones des parcelles en points spatiaux (centroïdes)
train.points <- SpatialPointsDataFrame(gCentroid(train.plots, byid=TRUE),
                                         data.frame(author=train.plots$author,
                                         ccov=train.plots$ccov,
                                         img_date=as.Date(train.plots$img_date))

# Pour la calibration de l'image 2018, sélectionner uniquement les points
# d'entraînement dont la date de l'image GoogleEarth est entre le 1.1.2017 et le 31.12
train.points <- train.points[!is.na(train.points$img_date) &
                               train.points$img_date > as.Date("2017-01-01") &
                               train.points$img_date <= as.Date("2019-12-31"), ]

# Illustrer la répartition des observations
pdf(paste0(REF.DIR, "/training-pts_2018.pdf"))
plot(train.points)
dev.off()

# Extraire les valeurs d'image pour les points d'entraînement (en parallèle)
registerDoParallel(CORES)
train.points <- foreach(i=1:length(ref.images), .combine=rbind) %dopar% {
  pts <- raster::extract(ref.images[[i]], train.points, sp=TRUE)
  pts <- raster::extract(bioclim[[i]], pts, sp=TRUE)
  pts$image <- names(ref.images[i])
  pts[, c("author", "image", "ccov", SSTS.BANDS, SSTS.BIOCLIM)]
}

# Supprimer les lignes avec des NA
train.points <- train.points[!is.na(rowSums(train.points@data[, -(1:2)])), ]

# Réorganiser le tableau des attributs (F10: forêt/non-forêt à 10% / F30: à 30%)
train.points@data <- cbind(train.points@data[,c("image", "ccov")],
                            F10=cut(train.points$ccov,
                                     breaks=c(0.0,0.1,1.0), labels=c(NONFOR, FOREST),
                                     right=FALSE, include.lowest=TRUE),
                            F30=cut(train.points$ccov,
                                     breaks=c(0.0,0.3,1.0), labels=c(NONFOR, FOREST),
                                     right=FALSE, include.lowest=TRUE),
                            train.points@data[,c(SSTS.LSBANDS, SSTS.BIOCLIM)])]

# Sélection des variables explicatives -----
cov.varsel <- rfe(y = train.points@data[train.points$image=="p193", "ccov"],
                     x = train.points@data[train.points$image=="p193", PREDICTORS],
                     sizes = c(4, 6, 8, 10),
                     rfeControl = rfeControl(
                       functions = rfFuncs,      # utiliser RandomForest

```

```

        method  = "repeatedcv",    # validation croisée
        number = 10,             # 10-fold
        repeats = 3)            # 3 répétitions

print(cov.varsel)
predictors(cov.varsel)
plot(cov.varsel, type=c("g", "o"))

# Carte de la couverture des houppiers pour 2018 (TODO) ----

set.seed(RSEED)
p193.2018.cov <- classify.image(image      = load.image("/p193/p193_2018_m.tif"),
                                    bioclim     = bioclim[["p193"]],
                                    filename   = paste0(REF.DIR, "/p193_2018_COV_R.tif"),
                                    train.pts  = train.points[train.points$image == "p193",
                                    pred       = PREDICTORS,
                                    type       = "treecover",
                                    crossval   = TRUE,
                                    ncores     = 32)

# observé vs. prédit
pdf(paste0(REF.DIR, "/p193_2018_COV_R.pdf"))
plot(p193.2018.cov[["model"]]$y, p193.2018.cov[["model"]]$predicted, xlim=c(0,1), ylim=c(0,1),
     main="Couverture houppier p193", xlab="Observation", ylab="Prédiction")
abline(0,1)
dev.off()

# Cartes de référence du couvert forestier 2018 ----

# Chemin WRS p193
set.seed(RSEED)
classify.image(image      = load.image("/p193/p193_2018_m.tif"),
               bioclim     = bioclim[["p193"]],
               filename   = paste0(REF.DIR, "/FC", COV.FC, "/p193_2018_FC", COV.FC, "_R.tif"),
               train.pts  = train.points[train.points$image == "p193", paste0("F", COV.FC, "_R.tif")],
               pred       = PREDICTORS,
               prob       = TRUE,
               crossval   = TRUE,
               ncores     = 32)

# Chemins WRS p192 and p194, utilisant p193 pour calibration
set.seed(RSEED)
registerDoParallel(CORES)
foreach(path=c("p192", "p194")) %dopar% {  # traitement en parallèle
  train.pts <- train.points[train.points$image == path, paste0("F", COV.FC)]
  classify.image(image      = load.image(paste0("/", path, "/", path, "_2018_m.tif")),
                 bioclim     = bioclim[[path]],
                 filename   = paste0(REF.DIR, "/FC", COV.FC, "/", path, "_2018_FC", COV.FC, "_R.tif"),
                 train.pts  = train.pts,
                 # calibration avec carte p193 ...
}

```

```

    cal.map    = raster(paste0(REF.DIR, "/FC", COV.FC, "/p193_2018_FC", COV.
n.cal.map = max(2000, nrow(train.pts)/CAL.RATIO), # ... avec au moins
preds      = PREDICTORS,
mask       = TGO,
prob       = TRUE,
n.cores    = 32)
}

# Cartes de référence du couvert forestier 2003 ----

# Chemin WRS p193
set.seed(RSEED)
classify.image(image      = load.image("/p193/p193_2003_m.tif"),
               bioclim    = bioclim[["p193"]],
               filename   = paste0(REF.DIR, "/FC", COV.FC, "/p193_2003_FC", COV.FC, "_R.t
# sur base de la carte de référence 2018
ref.map    = raster(paste0(REF.DIR, "/FC", COV.FC, "/p193_2018_FC", COV.FC,
n.ref.map = 2 * SAMPLE.RATIO * N.PIXELS[["p193"]],
preds      = PREDICTORS,
mask       = TGO,
n.cores    = 32)

# Chemins WRS p192 and p194, utilisant p193 pour calibration
set.seed(RSEED)
registerDoParallel(CORES)
foreach(path=c("p192", "p194")) %dopar% { # traitement en parallèle
  classify.image(image      = load.image(paste0("/", path, "/", path, "_2003_m.tif")),
                 bioclim    = bioclim[[path]],
                 filename   = paste0(REF.DIR, "/FC", COV.FC, "/", path, "_2003_FC", COV.F
# sur base de la carte de référence 2018 ...
ref.map    = raster(paste0(REF.DIR, "/FC", COV.FC, "/", path, "_2018_FC"
n.ref.map = 2 * (1 - CAL.RATIO) * SAMPLE.RATIO * N.PIXELS[[path]],
# ... et calibration avec carte p193 ...
cal.map    = raster(paste0(REF.DIR, "/FC", COV.FC, "/p193_2003_FC", COV.
n.cal.map = 2 * CAL.RATIO * SAMPLE.RATIO * N.PIXELS[[path]],
preds      = PREDICTORS,
mask       = TGO,
n.cores    = 32)
}

# Cartes du couvert forestier brutes pour toutes les dates ----

# Chemin WRS p193
set.seed(RSEED)
registerDoParallel(CORES)
foreach(file=dir(paste0(LANDSAT.DIR, "/p193"), pattern="\_\_[:digit:]]+\_\_m\_\_.tif")) %do
  classify.image(image      = load.image(paste0("/p193/", file)),
                 bioclim    = bioclim[["p193"]],
                 filename   = paste0(RAW.DIR, "/FC", COV.FC, "/p193/", sub("\_\_m\_\_.tif$",

```

```

        ref.map    = raster(paste0(REF.DIR, "/FC", COV.FC, "/p193_2003_FC", COV.
        n.ref.map = SAMPLE.RATIO * N.PIXELS[["p193"]],
        preds     = PREDICTORS,
        mask      = TGO,
        n.cores   = 6)
}

# fusionner les deux tuiles p193_1990
merge(raster(paste0(RAW.DIR, "/FC", COV.FC, "/p193/p193_1990_1_F", COV.FC, "r.tif")),
       raster(paste0(RAW.DIR, "/FC", COV.FC, "/p193/p193_1990_2_F", COV.FC, "r.tif")),
       filename=paste0(RAW.DIR, "/FC", COV.FC, "/p193/p193_1990_F", COV.FC, "r.tif"),
       format="GTiff", datatype="INT2U", overwrite=TRUE)

# Chemins WRS p192 and p194, utilisant ...
set.seed(RSEED)
registerDoParallel(CORES)
foreach(file=c(dir(paste0(LANDSAT.DIR, "/p192"), pattern="\_\_[:digit:]"]+\_\_m\_\_.tif"),
        dir(paste0(LANDSAT.DIR, "/p194"), pattern="\_\_[:digit:]"]+\_\_m\_\_.tif")))
path <- sub("\_\_.*", "", file)
# ... carte p193 pour la calibration (s'il existe, pour la même année)
if(file.exists(paste0(RAW.DIR, "/FC", COV.FC, "/p193/", sub("\_\_m\_\_.tif$", paste0("_F",
cal.map   <- raster(paste0(RAW.DIR, "/FC", COV.FC, "/p193/", sub("\_\_m\_\_.tif$", paste0("_F",
n.cal.map <- CAL.RATIO * SAMPLE.RATIO * N.PIXELS[[path]]
}) else {
  cal.map <- NULL
  n.cal.map <- NULL
}
classify.image(image      = load.image(paste0("/", path, "/", file)),
               bioclim   = bioclim[[path]],
               filename  = paste0(RAW.DIR, "/FC", COV.FC, "/", path, "/", sub("\_\_m\_\_.tif$",
               ref.map    = raster(paste0(REF.DIR, "/FC", COV.FC, "/", path, "_2003_FC",
               n.ref.map = (1 - CAL.RATIO) * SAMPLE.RATIO * N.PIXELS[[path]],
               cal.map    = cal.map,
               n.cal.map = n.cal.map,
               preds     = PREDICTORS,
               mask      = TGO,
               n.cores   = 6)
}

# Fusionner les cartes des chemins p192, p193 et p194 pour dates clés ... -----
for(year in YEARS.REF) {
  merge(mask(crop(brick(paste0(RAW.DIR, "/FC", COV.FC, "/p193/p193_", year, "_F", COV.FC,
  mask(crop(brick(paste0(RAW.DIR, "/FC", COV.FC, "/p192/p192_", year, "_F", COV.FC,
  mask(crop(brick(paste0(RAW.DIR, "/FC", COV.FC, "/p194/p194_", year, "_F", COV.FC,
  filename=paste0(RAW.DIR, "/FC", COV.FC, "/TGO/TGO_", year, "_F", COV.FC, "r.tif"))
}

# ... et le cartes de référence 2018 et 2003

```

```

for(map in c(paste0("2018_FC", COV.FC, "_R"),
            paste0("2018_FC", COV.FC, "_R_prob"),
            paste0("2003_FC", COV.FC, "_R"))){
  merge(mask(crop(brick(paste0(REF.DIR, "/FC", COV.FC, "/p193_", map, ".tif")), TGO), TGO,
             mask(crop(brick(paste0(REF.DIR, "/FC", COV.FC, "/p192_", map, ".tif")), TGO), TGO,
             mask(crop(brick(paste0(REF.DIR, "/FC", COV.FC, "/p194_", map, ".tif")), TGO), TGO,
             filename=paste0(REF.DIR, "/FC", COV.FC, "/TGO_", map, ".tif"), overwrite=TRUE)
}

```

0.4.1.2 Nettoyage des cartes brutes

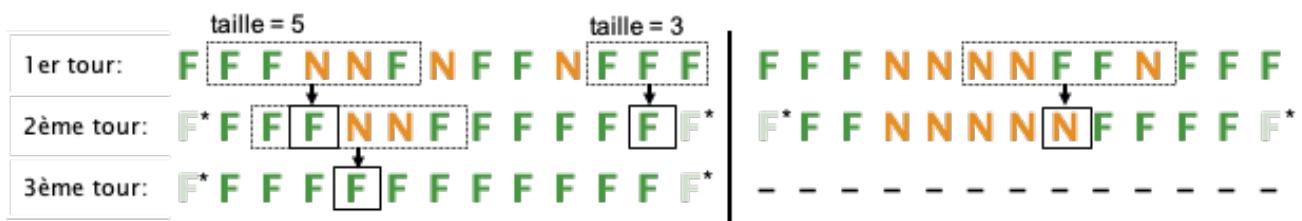
Les séries temporelles des cartes forêt/non-forêt brutes sont **lissées et filtrées pixel par pixel** pour les raisons suivantes:

- Remplissage des **données manquantes** (nuages, ombres, L7 SLC-off),
- Lissage des **bruits** (pixels qui changent entre forêt et non-forêt plusieurs fois)
- Filtrer la **régénération temporaire par les fourrés**, observée sur les jachères.

1. La première étape, est la **remplissage des données manquantes**: les données manquantes entre deux observations de forêt deviennent forêt, celles entre deux observations de non-forêt deviennent non-forêt.

... F x F ... → ... F F F ... | ... N x x N ... → ... N N N N ...

2. Ensuite, une **fenêtre coulissante de taille 5** est appliquée, c'est-à-dire qu'une observation est attribuée à la classe qui est observé le plus fréquemment dans la fenêtre qui inclut les deux observations précédentes et les deux suivantes. Pour l'évaluation de la deuxième et de l'avant-dernière observation, une fenêtre coulissante de taille 3 est appliquée. **La première et la dernière observation ne sont pas ajustées, il faut les ignorer dans la suite.** Les observations manquantes sont remplis dans la mesure possible. Toute la procédure est répétée jusqu'à ce qu'il n'y a plus de changements.



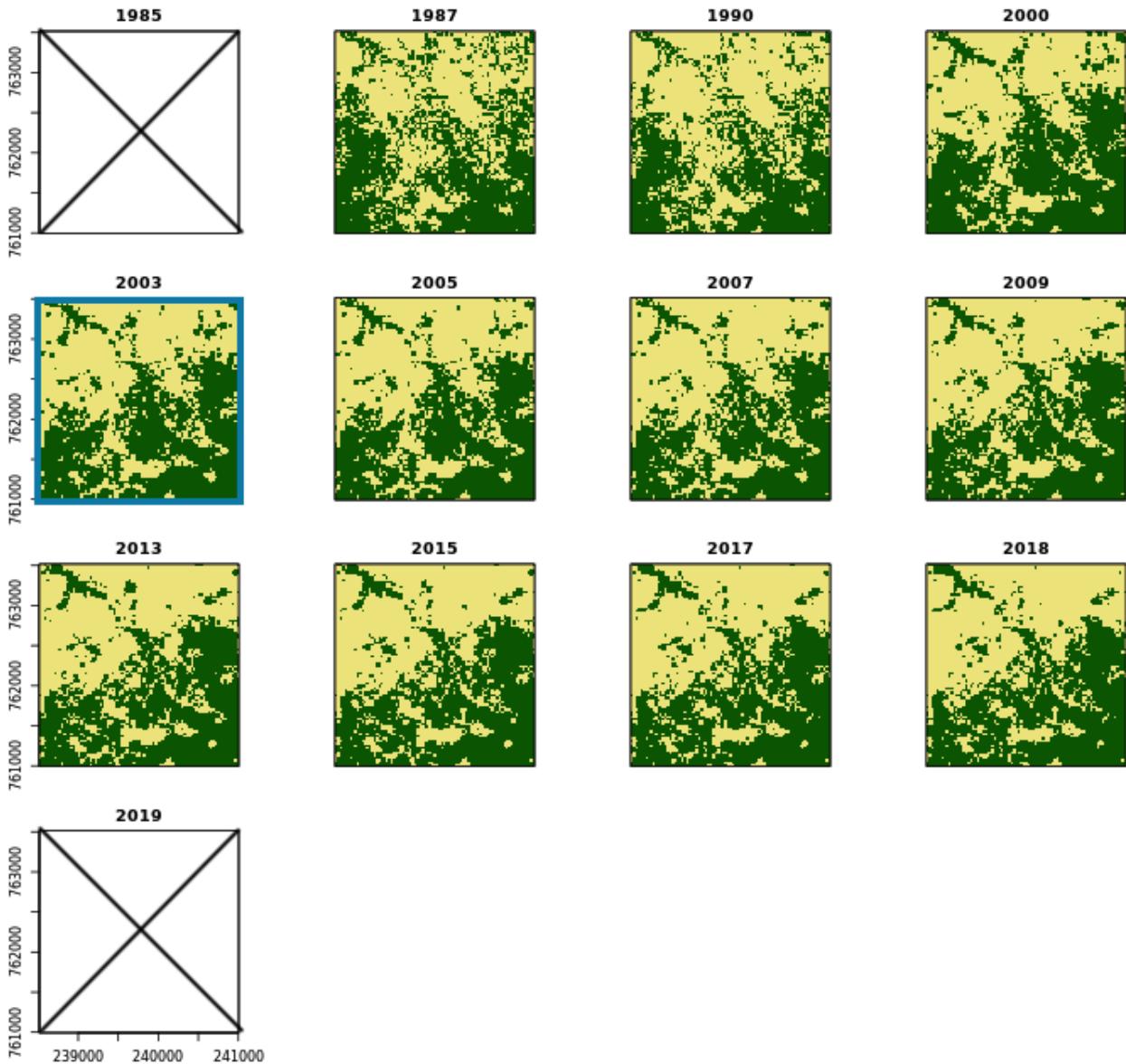
3. Les années manquantes sont ajoutées pour créer une série annuelle. Les observations manquantes sont remplacées par la classe de l'observation précédente. Lorsque les observations initiales sont manquantes, elles sont remplacées par la classe de l'observation suivante. **Dans les situations de régénération (forêt suit non-forêt), les 9 premières années sont marquées comme "reboisement potentiel". Si la forêt disparaît à nouveau après moins de 10 ans, les observations sont remplacées par la classe non-forêt. Si la régénération reste, elle est considérée comme un reboisement après 10 ans.**

F	x	x	N	→	F	F	F	N	
x	x	N	N	→	N	N	N	N	
N N N F F F F F N N N N				→	N				
N N F F F F F F F F				→	N N P P P P P P P P F				

Après la nettoyage des séries temporelles pixel par pixel, **une nettoyage spatiale** est réalisé pour éliminer les surfaces forestières < 0,5 hectares. Pour cela, une carte est créée de tous les pixels qui étaient une fois observées comme forêt sur toute la série des cartes. Sur cette carte, toutes les surfaces forestières ayant moins de 6 pixels liés (soit moins de 0,54 hectares) sont éliminées. Ce masque forestier est ensuite appliqué à toutes les cartes forêt/non-forêt de la série. Cela signifie que **seuls les pixels forestier qui ont fait partie d'une surface forestière $\geq 0,54$ hectares dans la série des cartes sont retenus comme forêt.**

0.4.1.2.1 Example

La figure suivante montre les **cartes forêt/non-forêt après la nettoyage temporelle et spatiale** pour une région au sud de Kpalimé (voir série des cartes brutes pour comparaison).



0.4.1.2.2 Script R: 03_NRF-MRV/01_MCF/_src/02_clean-fc-maps.R

```
#####
# 02_clean-fc-maps.R: nettoyage des cartes brutes du couvert forestier
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables =====
COV.FC      <- 30

RAW.DIR <- DIR.MRV.MCF.RAW
CLN.DIR <- DIR.MRV.MCF.CLN
```

```

# Définitions des fonctions =====

# Nettoyage temporel d'une série d'images -----
#
# @param path  Chemin WRS
#
# @return      --
#

clean.temporal <- function(path) {

  # Préparation des images ----

  # Charger les cartes brutes
  maps <- stack(dir(paste0(RAW.DIR, "/FC", COV.FC, "/", path),
                    pattern=".*\[[\:\d:]\]{4}\_\_F.*\_\_.tif$", full.names=TRUE))
  map.names <- sub("r$", "", names(maps))
  # Noms de cartes à utiliser dans les colonnes de la matrice
  map.cols <- sub(paste0(path, "\_\_"), "X", sub("\_\_[\:\alnum:]+$","M", map.names))
  # Names à utiliser pour les années sans carte
  no.map.cols <- paste0("X", YEARS.ALL[!YEARS.ALL %in% gsub("[\:\alpha:]", "", map.cols)])
  # Joindre et ordonner les noms de colonnes
  col.order <- c(map.cols, no.map.cols)[order(c(map.cols, no.map.cols))]
  # Convertir les cartes en matrice (une carte par colonne, prend du temps)
  maps.values <- values(maps)
  colnames(maps.values) <- map.cols

  # Nettoyage parallèle des trajectoires des pixels ----

  # Définir des sous-ensembles des pixels (lignes) pour le traitement parallèle
  nsubsets <- CORES
  subsets <- c(0, floor((1:nsubsets)*(nrow(maps.values)/nsubsets)))
  # Traitement en parallèle
  registerDoParallel(CORES)
  maps.values.clean <- foreach(i=1:nsubsets, .combine=rbind) %dopar% {

    # 0. obtenir un sous-ensemble à partir de maps.values
    val <- maps.values[(subsets[i]+1):subsets[i+1], ]
    # mettre tout en NA qui n'est pas de la forêt ou non-forêt
    val[!(is.na(val) | val %in% c(FOREST,NONFOR))] <- NA

    # 1. Supprimer les NA isolées ----
    str <- apply(val, 1, paste, collapse="") # convertir en chaînes de caractères
    str.c <- gsub("NA", "9", str)           # remplacer NA avec 9
    # Nettoyer jusqu'à la convergence
    while(!identical(str, str.c)) {
      str <- str.c
      # Mettre NA dans la classe correspondante (forêt ou non-forêt)
      str.c <- gsub(paste0("^(.*)", NONFOR, ")9(9*", NONFOR, ".*)$"), paste0("\1", NONFO

```

```

str.c <- gsub(paste0("^(.*", FOREST, ")9(9*", FOREST, ".*)$"), paste0("\\"1", FOREST))
}

# Reconvertir en matrice numérique
val <- matrix(as.numeric(unlist(strsplit(str.c, ""))), ncol=ncol(val), byrow=TRUE)
val[val==9] <- NA # remplacer les 9 avec NA
colnames(val) <- map.cols

# 2. nettoyer les trajectoires avec fenêtre coulissante ----
val.c <- val
val.o <- val[] <- 0
iter <- 0
# Nettoyer jusqu'à la convergence
while(!identical(val, val.c) & !identical(val.o, val.c)) {
  iter <- iter+1
  message(" -Clean modal: iteration ", iter, " ... ", appendLF = FALSE)
  val.o <- val # valeurs actuelles -> anciennes valeurs
  val <- val.c # valeurs nettoyées -> valuers actuelles
  # 3ème - 3ème l'année dernière : fenêtre modale taille 5
  for(l in 3:(ncol(val)-2)) {
    val.c[,l] <- apply(val[, (l-2):(l+2)], 1, modal, na.rm=TRUE, ties='NA')
  }
  # 2e et 2e l'année dernière : fenêtre modale taille 3
  for(l in c(2, ncol(val)-1)) {
    val.c[,l] <- apply(val.c[, (l-1):(l+1)], 1, modal, na.rm=TRUE, ties='NA')
  }
  message("done")
}
val <- val.c # valeurs nettoyées -> valuers actuelles

# 3. nettoyage final regexp ----
# ajouter des années sans observation (cartes)
val <- cbind(val, matrix(nrow=nrow(val),
                        ncol=length(no.map.cols), dimnames=list(NULL, no.map.cols)))
# ordonner correctement
val <- val[, col.order]
str <- apply(val, 1, paste, collapse="")
str.c <- gsub("NA", "9", str) # convertir en chaînes de caractères
# remplacer NA avec 9
# Nettoyer jusqu'à la convergence
while(!identical(str, str.c)) {
  str <- str.c
  # 3.1 remplacer NA par la classe précédente
  str.c <- gsub(paste0(NONFOR, "9"), paste0(NONFOR, NONFOR), str.c)
  str.c <- gsub(paste0(FOREST, "9"), paste0(FOREST, FOREST), str.c)
}
str <- ""
# Nettoyer jusqu'à la convergence
while(!identical(str, str.c)) {
  str <- str.c
  # 3.2 remplacer NA par la classe suivante
  str.c <- gsub(paste0("9", NONFOR), paste0(NONFOR, NONFOR), str.c)
}

```

```

    str.c <- gsub(paste0("9", FOREST), paste0(FOREST, FOREST), str.c)
}
str <- ""
# Nettoyer jusqu'à la convergence
while(!identical(str, str.c)) {
  str <- str.c
  # 3.3 suppression des 1 isolés jusqu'à une longueur de 10 (régénération qui est à
  str.c <- gsub(paste0("^(*", NONFOR, ")", FOREST, "(", FOREST, "{0,9}", NONFOR, ".*",
                      paste0("\\"1", NONFOR, "\\"2"), str.c)
}
str <- ""
# Nettoyer jusqu'à la convergence
while(!identical(str, str.c)) {
  str <- str.c
  # 3.4 considérer la régénération seulement comme une forêt à partir de 10 ans
  # avant qu'elle ne soit considérée comme une régénération potentielle PREGEN
  str.c <- gsub(paste0("^(*", NONFOREST, PREGEN, "{0,8})", FOREST, "(.*)$"),
                paste0("\\"1", PREGEN, "\\"2"), str.c)
}
str <- ""
# Reconvertir en matrice numérique
val <- matrix(as.numeric(unlist(strsplit(str.c, ""))), ncol=ncol(val), byrow=TRUE)
val[val==9] <- NA                                     # remplacer les 9 avec NA
colnames(val) <- col.order
val[,grepl("M$", colnames(val))]                     # et extraire les données pour les années
}

# Sauvegarder le résultat -----
values(maps) <- maps.values.clean
# supprimer la première et la dernière carte "non nettoyée"
writeRaster(dropLayer(maps, c(1,nlayers(maps))),
            filename=paste0(CLN.DIR, "/FC", COV.FC, "/", path, "/", map.names[2:(nlayers(maps))],
            bylayer=TRUE, format="GTiff", datatype="INT2U", overwrite=TRUE))

# écrire des trajectoires dans un fichier texte
maps.strings <- apply(maps.values.clean, 1, paste, collapse="")
sink(paste0(CLN.DIR, "/FC", COV.FC, "/", path, "/Trajectories.txt"))
print(table(maps.strings))
sink()

}

# Nettoyage spatiale d'une série d'images -----
#
# @description Supprimer les pixels de forêt/déforestation/régénération qui n'ont
#             JAMAIS fait partie de "forêt > 0,5 ha" depuis 2003
#
# @param maps           Série des cartes
# @param exclude        Cartes à ignorer

```

```

# @param size           Nombre minimal de pixels connectés
# @param connectedness Nombre de pixels qui comptent comme connectés
#
# @return               --
#
#
# clean.spatial.forest <- function(maps, exclude = NULL, size=6, connectedness=8){

tmp1 <- tempfile(pattern = "", fileext = ".tif")
tmp2 <- tempfile(pattern = "", fileext = ".tif")

fcc.map <- apply(maps[[-exclude]][], 1, paste, collapse="")
onceforest.map <- raster(maps)
onceforest.map[] <- NA

# créer une carte "forêt une fois"
onceforest.map[grep1(paste0("^", NONFOR, "*$"), fcc.map)] <- NONFOR
onceforest.map[grep1(paste0("^.*", FOREST, ".$"), fcc.map)] <- FOREST
onceforest.map[grep1(paste0("^.*", PREGEN, ".$"), fcc.map)] <- FOREST
writeRaster(onceforest.map, tmp1)

# supprimer les parcelles de forêt isolées < xy ha
system(paste0("gdal_sieve.py -st ", size, " -", connectedness, " -nomask ", tmp1, " "))
onceforest.map.clean <- raster(tmp2)
onceforest.map.clean[onceforest.map.clean == -2147483648] <- NA

# masquer les cartes avec cette carte forestière nettoyée
maps.clean <- mask(maps, onceforest.map.clean, maskvalue=NONFOR, updatevalue=NONFOR)
writeRaster(maps.clean, filename=paste0(CLN.DIR, "/FC", COV.FC, "/TGO/", names(maps.cl
bylayer=TRUE, format="GTiff", datatype="INT2U", overwrite=TRUE))

}

# COMMENCER LE TRAITEMENT ##### #####
# changer l'étendue de p192_2019 à l'étendue des autres images p192 -----
# TODO : à faire déjà dans 01_SSTS/01_data/_src/prep-Landsat.R
extend(brick(paste0(RAW.DIR, "/FC", COV.FC, "/p192/p192_2019_F", COV.FC, "r.tif")), rast
filename=paste0(RAW.DIR, "/FC", COV.FC, "/p192/p192_2019_F", COV.FC, "rt.tif"), f
file.rename(paste0(RAW.DIR, "/FC", COV.FC, "/p192/p192_2019_F", COV.FC, "rt.tif"), paste

# Nettoyage temporel des chemins -----
for(path in c("p192", "p193", "p194")) {
  clean.temporal(path)
}

# Fusionner les cartes p192, p193 et p194 pour les dates conjointes -----
for(year in YEARS.JNT) {

```

```

merge(mask(crop(brick(paste0(CLN.DIR, "/FC", COV.FC, "/p193/p193_"), year, "_F", COV.FC,
  mask(crop(brick(paste0(CLN.DIR, "/FC", COV.FC, "/p192/p192_"), year, "_F", COV.FC,
  mask(crop(brick(paste0(CLN.DIR, "/FC", COV.FC, "/p194/p194_"), year, "_F", COV.FC,
  filename=paste0(CLN.DIR, "/FC", COV.FC, "/TGO/TGO_"), year, "_F", COV.FC, "c.tif"))
})

# Spatial cleaning of results (only from 2003 onwards) -----
# exclure la première couche (1987) pour la création du masque forêt/non-forêt
clean.spatial.forest(maps = stack(dir(paste0(CLN.DIR, "/FC", COV.FC, "/TGO")), pattern =
  exclude = c(1), size = 6, connectedness = 8)

```

0.4.1.3 Validation des cartes

La validation des cartes forêt/non-forêt se fait à l'aide de parcelles de validation sur lesquelles l'**occupation du sol (forêt/terre boisée/non-forêt)** a été déterminée par des photo-interprètes sur la base d'images Landsat pour les années 1987, 2003, 2015 et 2018. Pour ces parcelles, dans un premier temps, les classes correspondantes sont lues à partir des cartes. Ensuite, des matrices d'erreur sont générées, celles de la classification forêt/non-forestière pour les différentes années, ainsi que celles de l'évolution de la couverture terrestre sur différentes périodes. Ces matrices d'erreurs constituent la base de l'analyse de la précision des cartes.

0.4.1.3.1 Example

Le tableau suivant montre la **matrice d'erreur des transitions de l'occupation du sol 2003 – 2018**. Dans les colonnes sont les classes attribuées par les photo-interprètes, dans les lignes les classes selon les cartes forêt/non-forêt nettoyées.

	xFxF	xFxN	xNxF	xNxN
xFxF	536	52	80	105
xFxN	36	80	3	52
xNxF	23	0	73	29
xNxN	35	84	50	1175

0.4.1.3.2 MCF/03_validate-fc-maps.R

```

#####
# 03_validate-fc-maps.R: validation des cartes forêt/non-forêt nettoyées
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables =====

```

```

COV.FC           <- 30

REF.DIR <- DIR.MRV.MCF.REF
RAW.DIR <- DIR.MRV.MCF.RAW
CLN.DIR <- DIR.MRV.MCF.CLN
VAL.DIR <- DIR.MRV.MCF.VAL
TPS.DIR <- DIR.SST.BDD.TPS
VPS.DIR <- DIR.SST.BDD.VPS

ct <- list()      # liste des matrices d'erreurs (tableaux de confusion)

# Préparations =====

# Charger cartes et points d'entraînement ----

# Cartes brutes et cartes nettoyées
maps <- stack(c(paste0(RAW.DIR, "/FC", COV.FC, "/TGO/TGO_", YEARS.REF, "_F", COV.FC, "r",
                     paste0(CLN.DIR, "/FC", COV.FC, "/TGO/TGO_", YEARS.REF, "_F", COV.FC, "c"),
                     names(maps) <- sub("TGO\\\_ ", "X", sub("\\_F[:digit:]]{2}", "", names(maps)))

# Points d'entraînement: Couverture des houppiers ...
train.plots <- readOGR(paste0(TPS.DIR, "/COV_parcelles.shp"))
train.points <- SpatialPointsDataFrame(gCentroid(train.plots, byid=TRUE), # polygons -
                                         data.frame(COV=train.plots$ccov)) # seulement C
# ... et classes forêt/non-forêt dans les cartes 2018
train.points <- raster::extract(maps[[c("X2018r", "X2018c")]], train.points, sp=TRUE)

# supprimer lignes avec NA
train.points <- train.points[rowSums(is.na(train.points@data)) == 0, ]

# charger points de validation ----

# Points de validation: classes forêt/non-forêt 1987, 2003, 2015, 2018 ...
val.plots <- readOGR(paste0(VPS.DIR, "/UOT_parcelles.shp"))
val.points <- SpatialPointsDataFrame(gCentroid(val.plots, byid=TRUE),
                                         data.frame(autho=sub("^.*\\/", "", val.plots$autho,
                                         V1987=as.numeric(substr(val.plots$lc_87,
                                         V2003=as.numeric(substr(val.plots$lc_03,
                                         V2015=as.numeric(substr(val.plots$lc_15,
                                         V2018=as.numeric(substr(val.plots$lc_18,

# ... et classes correspondantes dans les cartes
val.points <- raster::extract(maps, val.points, sp=TRUE)

# supprimer lignes avec NA

```

```
val.points <- val.points[rowSums(is.na(val.points@data)) == 0, ]
```

Ajustements terres boisées (classe 2) -----

```
if(COV.FC == 30) {
  # terre boisée -> non-forêt
  val.points$V1987[val.points$V1987 == 2] <- NONFOR
  val.points$V2003[val.points$V2003 == 2] <- NONFOR
  val.points$V2015[val.points$V2015 == 2] <- NONFOR
  val.points$V2018[val.points$V2018 == 2] <- NONFOR
}
if(COV.FC == 10) {
  # terre boisée -> forêt
  val.points$V1987[val.points$V1987 == 2] <- FOREST
  val.points$V2003[val.points$V2003 == 2] <- FOREST
  val.points$V2015[val.points$V2015 == 2] <- FOREST
  val.points$V2018[val.points$V2018 == 2] <- FOREST
}
```

Ajustements nuages/ombre (classe 4) -----

```
val.points$V1987r <- val.points$V1987c <- val.points$V1987;
val.points$V2003r <- val.points$V2003c <- val.points$V2003;
val.points$V2015r <- val.points$V2015c <- val.points$V2015;
val.points$V2018r <- val.points$V2018c <- val.points$V2018;
```

Prendre la classe dans les cartes brutes ...

```
val.points$V1987r[val.points$V1987 %in% c(2,4)] <- val.points$X1987r[val.points$V1987 %in% c(2,4)]
val.points$V2003r[val.points$V2003 %in% c(2,4)] <- val.points$X2003r[val.points$V2003 %in% c(2,4)]
val.points$V2015r[val.points$V2015 %in% c(2,4)] <- val.points$X2015r[val.points$V2015 %in% c(2,4)]
val.points$V2018r[val.points$V2018 %in% c(2,4)] <- val.points$X2018r[val.points$V2018 %in% c(2,4)]
```

... et les cartes nettoyées

```
val.points$V1987c[val.points$V1987 %in% c(2,4)] <- val.points$X1987c[val.points$V1987 %in% c(2,4)]
val.points$V2003c[val.points$V2003 %in% c(2,4)] <- val.points$X2003c[val.points$V2003 %in% c(2,4)]
val.points$V2015c[val.points$V2015 %in% c(2,4)] <- val.points$X2015c[val.points$V2015 %in% c(2,4)]
val.points$V2018c[val.points$V2018 %in% c(2,4)] <- val.points$X2018c[val.points$V2018 %in% c(2,4)]
```

Ajustements de la régénération potentielle ----

changer "régénération potentielle" à la classe identifiée dans les parcelles de vali

```
val.points$X1987c[val.points$X1987c == PREGEN] <- val.points$V1987c[val.points$X1987c == PREGEN]
val.points$X2003c[val.points$X2003c == PREGEN] <- val.points$V2003c[val.points$X2003c == PREGEN]
val.points$X2015c[val.points$X2015c == PREGEN] <- val.points$V2015c[val.points$X2015c == PREGEN]
val.points$X2018c[val.points$X2018c == PREGEN] <- val.points$V2018c[val.points$X2018c == PREGEN]
```

Non-forêt où "régénération potentielle" dans la carte et "terre boisée" dans les par

```
val.points$X1987c[val.points$X1987c == NONFOR] <- val.points$V1987c[val.points$X1987c == NONFOR]
```

```

val.points$X2003c[val.points$X2003c == NONFOR] <- val.points$V2003c[val.points$X2003c ==]
val.points$X2015c[val.points$X2015c == NONFOR] <- val.points$V2015c[val.points$X2015c ==]
val.points$X2018c[val.points$X2018c == NONFOR] <- val.points$V2018c[val.points$X2018c ==]

val.points@data <- mutate_all(val.points@data, as.factor)

# Matrices d'erreurs =====

# carte 2018 vs. carte référence 2018 ----

ref.map <- raster(paste0(REF.DIR, "/FC", COV.FC, "/TGO_2018_FC", COV.FC, "_R.tif"))
ct[["MAP_REF.18r"]] <- confusionMatrix(as.factor(maps$X2018r[]), as.factor(ref.map[]))

pdf(paste0(VAL.DIR, "/FC2018_vs_COV2018_FC", COV.FC, ".pdf"))
plot(factor(train.points$X2018r, labels=c("Forest", "Non-Forest")) ~ train.points$COV, x)
dev.off()

pdf(paste0(VAL.DIR, "/COV2018_vs_FC2018_FC", COV.FC, ".pdf"))
plot(train.points$COV ~ factor(train.points$X2018r, labels=c("Forest", "Non-Forest")), x)
dev.off()

# Map validation / confusion matrices ----

confMat <- function(xtrans, vtrans) {
  levels <- unique(c(xtrans, vtrans))
  return(confusionMatrix(factor(xtrans, levels=levels[order(levels)]), factor(vtrans, l
}

# val.points.bu <- val.points      # backup
val.points <- val.points[!val.points$author %in% c("7_Aklasson_TOLEBA", "8_Yawo_KONKO",

# check for authors whose validation points reduce precision
# print(confMat(paste0("x", tmp$X2003c, tmp$X2015c, tmp$X2018c),
#               paste0("x", tmp$VX2003, tmp$VX2015, tmp$VX2018))$overall)
#
# for(author in unique(tmp$author)) {
#   print(author)
#   print(confMat(paste0("x", tmp$X2003c[tmp$author != author], tmp$X2015c[tmp$author
#                           paste0("x", tmp$VX2003[tmp$author != author], tmp$VX2015[tmp$author
#   } }

for(t in c("r", "c")) {

  # confusion matrices for individual dates
  ct[[paste0("MAP_VAL.87", t)]] <- confMat(paste0(val.points[[paste0("X1987", t)]], val.
  ct[[paste0("MAP_VAL.03", t)]] <- confMat(paste0("x", val.points[[paste0("X1987", t)]], val.
}

```

```

ct[[paste0("MAP_VAL.15", t)]]           <- confMat(paste0("x", "x",
ct[[paste0("MAP_VAL.18", t)]]           <- confMat(paste0("x", "x",
                                         "x"))

# confusion matrices for 2-date transitions
ct[[paste0("MAP_VAL.87.03", t)]]         <- confMat(paste0(val.points[[paste0("X1987", t)]]))
ct[[paste0("MAP_VAL.03.15", t)]]         <- confMat(paste0("x", val.points[[paste0("X1987", t)]])
ct[[paste0("MAP_VAL.15.18", t)]]         <- confMat(paste0("x", "x", val.points[[paste0("X1987", t)]])
ct[[paste0("MAP_VAL.87.18", t)]]         <- confMat(paste0(val.points[[paste0("X1987", t)]])
ct[[paste0("MAP_VAL.03.18", t)]]         <- confMat(paste0("x", val.points[[paste0("X1987", t)]))

# confusion matrices for 3-date transition
ct[[paste0("MAP_VAL.03.15.18", t)]]      <- confMat(paste0("x", val.points[[paste0("X2003", t)]))

# confusion matrices for 4-date transition
ct[[paste0("MAP_VAL.87.03.15.18", t)]]    <- confMat(paste0(val.points[[paste0("X1987", t)]))

}

# Validation RapidEye map 2015 -----
rey <- raster(paste0(DATA.DIR, "/RapidEye/TGO_30m.tif"))
names(rey) <- "R2015"
val.points <- raster::extract(rey, val.points, sp=TRUE)

val.points$Rr2015 <- 3
val.points$Rr2015[val.points$R2015 %in% c(11, 12, 16, 18, 19)] <- 1
val.points$VR2015 <- val.points$V2015
val.points$VR2015[val.points$V2015 == 2] <- val.points$Rr2015[val.points$V2015 == 2]

ct[["REY_VAL.15"]]           <- confMat(paste0("x", "x", val.points$Rr2015,
                                             "x"))

# Write confusion tables -----
save(ct, file=paste0(FCC.VAL.DIR, "/FC", COV.FC, "_flex_ConfTab.RData"))

# write error matrices to Excel File
write.xlsx(lapply(ct, "[[", "table"),
           file=paste0(FCC.VAL.DIR, "/FC", COV.FC, "_flex_ConfTab.xlsx"),
           colnames=TRUE, overwrite=TRUE)

```

0.4.1.4 Analyse de la précision

La précision des cartes est déterminée selon la procédure de Olofsson et al. (2014). Le script utilisé est basé sur une implémentation des méthodes dans OpenForis.

Le calcul de la précision se fait sur la matrice d'erreur d'une part et les surfaces couvertes par les catégories correspondantes d'autre part. Avec les informations sur les surfaces, la matrice d'erreur est extrapolée en une **matrice d'erreur proportionnelle**, qui indique dans les cellules les proportions de la surface totale. Sur cette base les indices de précision sont calculés: concrètement la précision globale, le Kappa de Cohen, la précision du producteur et la précision de l'utilisateur, y compris la variance et l'intervalle de confiance à 95 %. D'autre part, la matrice d'erreur pro-

portionnelle est extrapolée à la superficie totale pour obtenir des estimations de superficie des différentes catégories, y compris les erreurs types et les intervalles de confiance.

0.4.1.4.1 Example

Le nombre de pixels cartographiés et matrice d'erreur des transitions de l'occupation du sol 2003 – 2018. Dans les colonnes sont les classes attribuées par les photo-interprètes, dans les lignes les classes selon les cartes forêt/non-forêt nettoyées.

	Pixels	xFxF	xFxN	xNx F	xNxN
xFxF	12 590 594	536	52	80	105
xFxN	2 509 971	36	80	3	52
xNx F	1 637 331	23	0	73	29
xNxN	46 599 650	35	84	50	1175

La matrice d'erreur proportionnelle aux surfaces cartographiées. La somme de tous les cellules est 1 (100%).

	xFxF	xFxN	xNx F	xNxN
xFxF	0.138	0.013	0.021	0.027
xFxN	0.008	0.019	0.001	0.012
xNx F	0.005	0	0.015	0.006
xNxN	0.019	0.046	0.027	0.643

Les **indices de précision globales** sont:

- Précision globale: 81.5% (\pm 1.5% intervalle de confiance)
- Cohen's Kappa: 59.3%

Indices de précision et le surfaces estimées par catégories (\pm intervalle de confiance à 95%)

	Surface cartes	Prop. cartes	Proportion ajustée	Surface ajustée	Précision utilisateur	Précision producteur
xFxF	1'133'153	0.20	0.17 \pm 0.01	969'621 \pm 54'104	69 \pm 0.03	0.81 \pm 0.03
xFxN	225'897	0.04	0.08 \pm 0.01	444'034 \pm 60'300	47 \pm 0.08	0.24 \pm 0.04
xNx F	147'360	0.03	0.06 \pm 0.01	363'320 \pm 50'705	8 \pm 0.09	0.24 \pm 0.04
xNxN	4'193'969	0.74	0.69 \pm 0.01	3'923'405 \pm 81'587	58 \pm 0.02	0.94 \pm 0.01

0.4.1.4.2 MCF/04_fc-maps-accuracy.R

```
#####
# NERF_Togo/FCC/7_fc-maps-accuracy.R: validate clean forest cover maps
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 20 May 2019

# based on OpenForis implementation of Olofsson et al. (2014), written by
# Antonia Ortmann, 20 October, 2014
# Source: https://github.com/openforis/accuracy-assessment/blob/master/Rscripts/error_1.R

VALSET <- "FC30_flex"

# Function for estimating accuracies -----
accuracy.estimate <- function(areas.map, error.matrix, filename=NULL, pixelsize=30^2/100)

# remove "x" from the category
colnames(error.matrix) <- rownames(error.matrix) <- gsub("x", "", colnames(error.matrix))

# match category order
maparea      <- areas.map[match(rownames(error.matrix), names(areas.map))]
ma           <- error.matrix
dyn          <- names(maparea)

# calculate the area proportions for each map class
aoi <- sum(maparea)
propmaparea <- maparea/aoi

# convert the absolute cross tab into a probability cross tab
ni. <- rowSums(ma)                      # number of reference points per map class
propma <- as.matrix(ma/ni. * as.vector(propmaparea))
propma[is.nan(propma)] <- 0             # for classes with ni. = 0

# estimate the accuracies now
OA <- sum(diag(propma))                # overall accuracy (Eq. 1 in Olofsson et al. 2014)
pe <- 0                                  # Agreement by chance ...
for (i in 1:length(dyn)) {
  pe <- pe + sum(propma[i,]) * sum(propma[,i])
}
K <- (OA - pe) / (1 - pe)               # ... for Cohen's Kappa
UA <- diag(propma) / rowSums(propma)    # user's accuracy (Eq. 2 in Olofsson et al. 2014)
PA <- diag(propma) / colSums(propma)    # producer's accuracy (Eq. 3 in Olofsson et al. 2014)

# estimate confidence intervals for the accuracies
V_OA <- sum(as.vector(propmaparea)^2 * UA * (1 - UA) / (ni. - 1), na.rm=T) # variance of overall accuracy
V_UA <- UA * (1 - UA) / (rowSums(ma) - 1) # variance of user's accuracy (Eq. 6 in Olofsson et al. 2014)
N.j <- array(0, dim=length(dyn))

# variance of producer's accuracy (Eq. 7 in Olofsson et al. 2014)
```

```

aftersumsign <- array(0, dim=length(dyn))
for(cj in 1:length(dyn)) {
  N.j[cj] <- sum(maparea / ni. * ma[, cj], na.rm=T)
  aftersumsign[cj] <- sum(maparea[-cj]^2 * ma[-cj, cj] / ni.[-cj] * (1 - ma[-cj, cj])
}
V_PA <- 1/N.j^2 * (
  maparea^2 * (1-PA)^2 * UA * (1-UA) / (ni.-1) +
  PA^2 * aftersumsign
)
V_PA[is.nan(V_PA)] <- 0

# proportional area estimation
propAreaEst <- colSums(propma)           # proportion of area (Eq. 8 in Olofsson et al.
AreaEst <- propAreaEst * sum(maparea) # estimated area

# standard errors of the area estimation (Eq. 10 in Olofsson et al. 2014)
V_propAreaEst <- array(0, dim=length(dyn))
for (cj in 1:length(dyn)) {
  V_propAreaEst[cj] <- sum(as.vector(propmaparea) * propma[, cj] - propma[, cj]^2)
}
V_propAreaEst[is.na(V_propAreaEst)] <- 0

# produce result tables

res <- list()
res$PREDICTED_PX      <- as.table(maparea)
res$ERROR_MATRIX       <- ma
res$ERROR_MATRIX_PROP <- round(propma, 3)
res$OVERALL_ACC        <- data.frame(accuracy=round(c(OA, K), 3),
                                         CI=round(c(1.96 * sqrt(V_OA), NA), 3),
                                         row.names=c("OA", "Kappa"))
res$CLASSES_ACC <- data.frame(maparea=round(maparea * pixelsize, 3)) # in ha
res$CLASSES_ACC$prop_maparea <- round(propmaparea, 3)
res$CLASSES_ACC$adj_proparea <- round(propAreaEst, 3)
res$CLASSES_ACC$CI_adj_proparea <- round(1.96 * sqrt(V_propAreaEst), 3)
res$CLASSES_ACC$adj_area <- round(propAreaEst * aoi * pixelsize, 3) # in ha
res$CLASSES_ACC$CI_adj_area <- round(1.96 * sqrt(V_propAreaEst) * aoi * pixelsize,
                                         3)
res$CLASSES_ACC$UA       <- round(UA, 3)
res$CLASSES_ACC$CI_UA    <- round(1.96 * sqrt(V_UA), 3)
res$CLASSES_ACC$PA       <- round(PA, 3)
res$CLASSES_ACC$CI_PA    <- round(1.96 * sqrt(V_PA), 3)

# write results to Excel File
if(!is.null(filename)) {
  write.xlsx(res,
             file=filename,
             col.names=TRUE, row.names=TRUE, overwrite=TRUE)
}

return(res)

```

```
}
```

```
# DO THE WORK -----
```

```
# load the error matrices (R object ct)
load(paste0(FCC.VAL.DIR, "/ConfTab_", VALSET, ".RData"))

# load the predictions and convert "potential regeneration (2)" to "non-forest (3)"
fc.2003 <- brick(paste0(FCC.CLN.DIR, "/FC30/TGO/TGO_2003_F30cf.tif")); fc.2003[fc.2003==
fc.2015 <- brick(paste0(FCC.CLN.DIR, "/FC30/TGO/TGO_2015_F30cf.tif")); fc.2015[fc.2015==
fc.2018 <- brick(paste0(FCC.CLN.DIR, "/FC30/TGO/TGO_2018_F30cf.tif")); fc.2018[fc.2018==

# create 3-date transition map
fcc <- 100 * fc.2003 + 10 * fc.2015 + 1 * fc.2018

# get pixel counts (takes time) and separate for different dates / transitions
freq <- table(fcc[])
tmp <- as.numeric(freq); names(tmp) <- names(freq); freq <- tmp

freq.03.15.18      <- c(freq["111"], freq["113"], freq["131"], freq["133"], freq["311"])
names(freq.03.15.18) <- c("111", "113", "131", "133", "311", "313", "331", "333"); freq.03.15.18 <- freq.03.15.18 / sum(freq.03.15.18)

freq.03.18          <- c(sum(freq["111"], freq["131"], na.rm=T), sum(freq["113"], freq["133"],
                           sum(freq["311"], freq["331"], na.rm=T), sum(freq["313"], freq["333"]))
names(freq.03.18) <- c("11", "13", "31", "33"); freq.03.18[is.na(freq.03.18)] <- 0

freq.03.15          <- c(sum(freq["111"], freq["113"], na.rm=T), sum(freq["131"], freq["133"],
                           sum(freq["311"], freq["313"], na.rm=T), sum(freq["331"], freq["333"]))
names(freq.03.15) <- c("11", "13", "31", "33"); freq.03.15[is.na(freq.03.15)] <- 0

freq.15.18          <- c(sum(freq["111"], freq["311"], na.rm=T), sum(freq["113"], freq["313"],
                           sum(freq["131"], freq["331"], na.rm=T), sum(freq["133"], freq["333"]))
names(freq.15.18) <- c("11", "13", "31", "33"); freq.15.18[is.na(freq.15.18)] <- 0

freq.03              <- c(sum(freq["111"], freq["131"], freq["113"], freq["133"], na.rm=T),
                           sum(freq["311"], freq["331"], freq["313"], freq["333"], na.rm=T))
names(freq.03) <- c("1", "3"); freq.03[is.na(freq.03)] <- 0

freq.15              <- c(sum(freq["111"], freq["311"], freq["113"], freq["313"], na.rm=T),
                           sum(freq["131"], freq["331"], freq["133"], freq["333"], na.rm=T))
names(freq.15) <- c("1", "3"); freq.15[is.na(freq.15)] <- 0

freq.18              <- c(sum(freq["111"], freq["311"], freq["131"], freq["331"], na.rm=T),
                           sum(freq["113"], freq["313"], freq["133"], freq["333"], na.rm=T))
names(freq.18) <- c("1", "3"); freq.18[is.na(freq.18)] <- 0

# create accuracy maps -----
```

```

accuracy.estimate(freq.18, ct$MAP_VAL.18c$table , filename=paste0(FCC.VAL.DIR, "/",
accuracy.estimate(freq.15, ct$MAP_VAL.15c$table , filename=paste0(FCC.VAL.DIR, "/",
accuracy.estimate(freq.03, ct$MAP_VAL.03c$table , filename=paste0(FCC.VAL.DIR, "/",
VALS

accuracy.estimate(freq.03.15, ct$MAP_VAL.03.15c$table , filename=paste0(FCC.VAL.DIR, "/",
accuracy.estimate(freq.15.18, ct$MAP_VAL.15.18c$table , filename=paste0(FCC.VAL.DIR, "/",
accuracy.estimate(freq.03.18, ct$MAP_VAL.03.18c$table , filename=paste0(FCC.VAL.DIR, "/",

accuracy.estimate(freq.03.15.18, ct$MAP_VAL.03.15.18c$table , filename=paste0(FCC.VAL.DI

# calculate forest cover, loss and gains -----
res <- data.frame(year      = c(2003,           2015,           2018),
                   total.ha = c(freq.03["1"], freq.15["1"], freq.18["1"]) * 30^2,
                   defor.ha = c(NA,           freq.03.15["13"], freq.15.18["13"]) * 30^2,
                   regen.ha = c(NA,           freq.03.15["31"], freq.15.18["31"]) * 30^2

# Example from Olofsson et al. (2014)
# -----
# ct <- matrix(data=c(66, 0, 5, 4,
#                   0, 55, 8, 12,
#                   1, 0, 153, 11,
#                   2, 1, 9, 313),
#               nrow=4,
#               byrow=TRUE,
#               dimnames = list(Prediction=c("FN", "NF", "FF", "NN"), Reference=c("FN",
#               #
# px <- c(FN=200000, NF=150000, FF=3200000, NN=6450000)
# 
# accuracy.estimate(px, ct)

```

0.4.1.5 Analyse des cartes

Les cartes forêt/non-forêt nettoyées de 1987 à 2018 sont utilisées pour créer des cartes des changements forestiers. En utilisant la fonction `fcc()` les pixels qui passent d'une situation “non-forêt” ou “régénération potentielle” à “régénération” sont enregistrés comme gains de surface forestière dans l'année correspondante, les pixels qui passent d'une situation “forêt” ou “régénération” à “non-forêt” sont enregistrés comme perte de surface forestière. Sur cette base, pour chaque année disponible dans la série, une **compilation des surfaces forestières, de la surface régénérée et de la régénération potentielle**, ainsi que des gains et pertes de forêts dans la période précédente est effectuée.

Le tableau de la superficie forestière sert à son tour de base pour déterminer les **changements annuelles de la surface forestière sur différentes périodes de temps** à l'aide de la fonction `fcc()`. En plus des changements annuelles absolues, les taux de changement correspondants sont calculés selon la formule suivante : $r = (1/(t_2 - t_1)) \times \ln(A_2/A_1)$ (Puyravaud, 2003).

La fonction `plot.fcc()` affiche graphiquement le tableau des surfaces forestières et montre com-

ment la surface forestière et la régénération se développent. La fonction `plot.fcc()` crée un diagramme en barres des gains et des pertes annuelles de la surface forestières sur certaines périodes de temps.

Enfin, des **cartes de la perte de forêts, de la régénération potentielle et de la régénération** sont créées, avec les années de changement observées comme valeurs de pixel.

0.4.1.5.1 Example

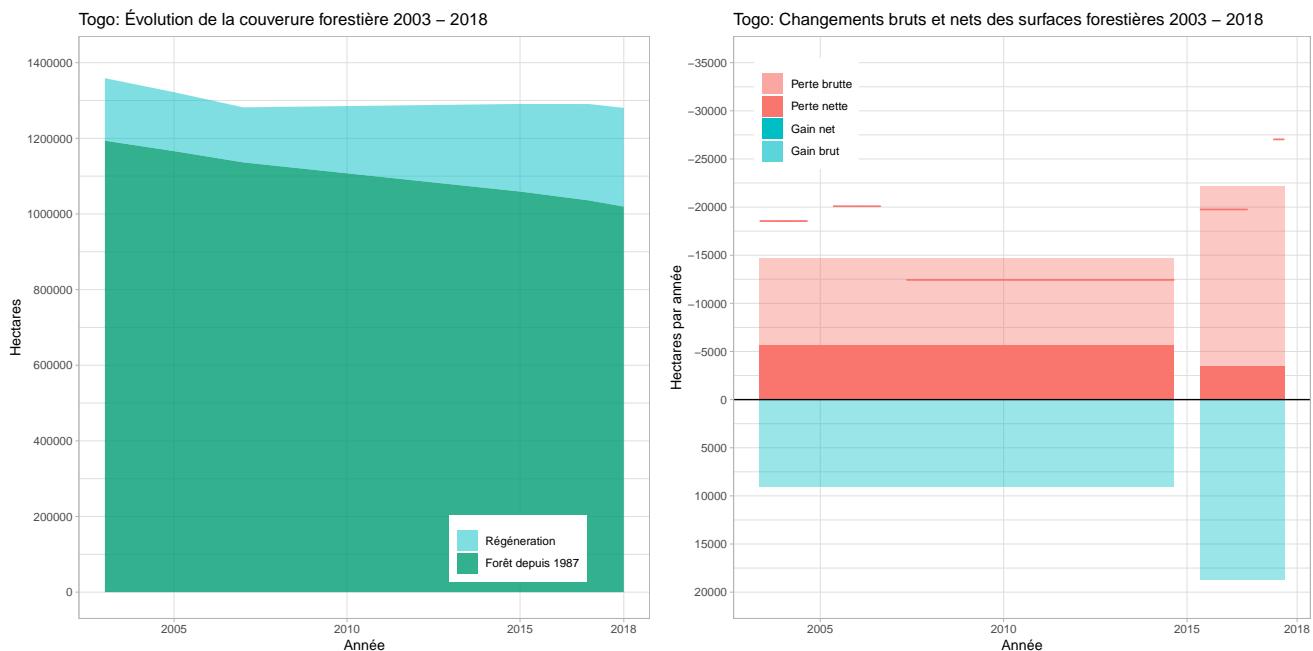
Tableau des superficies forestières pour le Togo La surface forestière initiale (existant depuis 1987), la superficie forestière secondaire (régénérée depuis 1987), la surface de la régénération potentielle (< 10 ans) et la déforestation et l'accroissement enregistrés dans la période précédente. Tous les chiffres sont exprimés en hectares.

Année	Surface f. totale	Surface f. initiale	Surface f. secondaire	Régén. potentielle	Pertes surface f.	Gains surface f.
1987	1'265'377	1'265'377	0	133'038	—	—
2003	1'359'051	1'193'731	165'320	90'972	-71'646	165'320
2005	1'321'963	1'166'156	155'807	122'718	-37'088	0
2007	1'281'909	1'136'373	145'536	161'787	-40'154	101
2015	1'290'948	1'059'301	231'647	156'237	-99'560	108'600
2017	1'290'615	1'035'724	254'891	169'171	-39'503	39'170
2018	1'280'513	1'019'489	261'024	188'715	-27'027	16'925

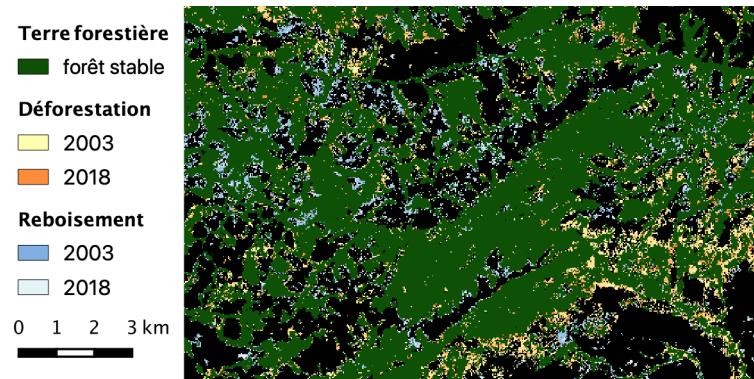
Changement annuelle de la superficie forestière au Togo, pour 2003 – 2015, 2015 – 2018 et toute la période 2003 – 2018, montrant les pertes et les gains brutes des forêts, et le changement net de la surface forestière en hectares par an et les taux correspondants en %/an.

Période	Pertes (ha/an)	Gains (ha/an)	Ch. net (ha/an)	Pertes (%/an)	Gains (%/an)	Ch. net (%/an)
2003 – 2015	-14'734	9'058	-5'675	-1.2	0.6	-0.4
2015 – 2018	-22'177	18'699	-3'478	-1.8	1.4	-0.3
2003 – 2018	-16'222	10'986	-5'236	-1.3	0.8	-0.4

Changement de la surface forestière du Togo 2003 – 2018 Changement de la superficie forestière (à gauche) ainsi que les gains et les pertes annuelles pour les périodes 2003 – 2015 et 2015 – 2018 (à droite).



Complexité de l'évolution de la superficie forestière en prenant l'exemple d'une petite partie de la région des Plateaux. Pertes de forêts 2003 – 2018 (jaune à orange) et croissance des forêts au cours de la même période (bleu à blanc). Le fond noir est non-forêt sur toute la période.



0.4.1.5.2 MCF/05_analyse-fc-maps.R

```
#####
# NERF_Togo/FCC/8_analyse-fc-maps.R: analyze clean forest cover maps
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 20 May 2019
```

```
COV.FC <- 30
```

```
# Function for building forest cover table -----
```

```
fc <- function(map, aoi=NULL) {
  if(!is.null(aoi)) map <- mask(crop(map, aoi), aoi)
```

```

# convert non-forest to 0
# potential regeneration to 1
# regeneration to 2
# forest to 3

tab.fc <- map[]
tab.fc[tab.fc == 3] <- 0
tab.fc[tab.fc == 1] <- 3
tab.fc[tab.fc == 2] <- 1

for(i in 2:ncol(tab.fc)) {
  tab.fc[!is.na(tab.fc[,i]) & tab.fc[,i] == 3 & !is.na(tab.fc[,i-1]) & tab.fc[,i-1] <
}

tab.fcc <- tab.fc[,1:ncol(tab.fc)-1]
tab.fcc[] <- 0
for(i in 1:ncol(tab.fcc)) {
  tab.fcc[tab.fc[,i] <= 1 & tab.fc[,i+1] >= 2, i] <- 1 # mark forest gain
  tab.fcc[tab.fc[,i] >= 2 & tab.fc[,i+1] <= 1, i] <- 2 # mark forest loss
}

dates <- as.numeric(substr(names(map), 2, 5))

fc <- data.frame(year      = dates,
                  initi.ha = colSums(tab.fc==3, na.rm=TRUE) * 30^2/10000,
                  secon.ha = colSums(tab.fc==2, na.rm=TRUE) * 30^2/10000,
                  poten.ha = colSums(tab.fc==1, na.rm=TRUE) * 30^2/10000)

fc$total.ha <- fc$initi.ha + fc$secon.ha
fc$defor.ha <- -c(colSums(tab.fcc==2, na.rm=TRUE) * 30^2/10000, NA)
fc$regen.ha <- c(colSums(tab.fcc==1, na.rm=TRUE) * 30^2/10000, NA)

return(fc)
}

fcc <- function(fc.tab, years=fc.tab$year) {

  for(i in 1:(length(years)-1)) {
    fc.tab$defor.ha.yr[fc.tab$year==years[i]] <- sum(fc.tab$defor.ha[fc.tab$year >= year
    fc.tab$regen.ha.yr[fc.tab$year==years[i]] <- sum(fc.tab$regen.ha[fc.tab$year >= year
    fc.tab$netch.ha.yr[fc.tab$year==years[i]] <- fc.tab$defor.ha.yr[fc.tab$year==years[i]

    fc.tab$defor.pc.yr[fc.tab$year==years[i]] <- round(100 * 1/(years[i+1]-years[i]) * 1
    fc.tab$regen.pc.yr[fc.tab$year==years[i]] <- round(100 * 1/(years[i+1]-years[i]) * 1
    fc.tab$netch.pc.yr[fc.tab$year==years[i]] <- round(100 * 1/(years[i+1]-years[i]) * 1
  }

  return(fc.tab)
}

```

```
}
```

Function for plotting evolution of forest cover -----

```
plot.fc <- function(fc, zone, filename=NULL) {

  if(zone=="TGO") {
    title     <- "Togo: Évolution de la couverture forestière 2003 - 2018"
    ylim      <- c(0, 1400000)
    ybreaks   <- seq(0, 1400000, 200000)
    ymbreaks  <- seq(0, 1400000, 100000)
  } else {
    title     <- paste0(zone, ": Évolution de la couverture forestière 2003 - 2018")
    ylim      <- c(0, 500000)
    ybreaks   <- seq(0, 500000, 100000)
    ymbreaks  <- seq(0, 500000, 50000)
  }

  if(!is.null(filename)) pdf(filename)

  print(
    fc %>% gather(variable, value, secon.ha, initi.ha) %>%
      ggplot(aes(x = year, y=value, fill=variable)) +
      geom_area(position=position_stack(reverse=TRUE)) +
      scale_fill_manual(name = NULL, breaks=c("secon.ha", "initi.ha"), values=c(alpha("#
        xlab("Année") + ylab("Hectares") + ggtitle(title) +
        scale_x_continuous(minor_breaks = NULL, breaks=c(1991, 2000, 2005, 2010, 2015, 201
        scale_y_continuous(breaks=ybreaks, minor_breaks = ymbreaks) + coord_cartesian(ylim
        theme_light() + theme(legend.position=c(0.6,0.2), legend.box = "horizontal", legen
    )
  )

  if(!is.null(filename)) dev.off()
}
```

Function for plotting forest cover change -----

```
plot.fcc <- function(fc, zone=NULL, breaks=NULL, filename=NULL) {

  my.fcc <- fcc(fc)
  fcc.breaks <- my.fcc

  if(!is.null(breaks)) fcc.breaks <- fcc(fc, breaks)[fc$year %in% breaks,]

  my.fcc$period <- c(my.fcc$year[2:nrow(my.fcc)] - my.fcc$year[1:(nrow(my.fcc)-1)], NA)
  my.fcc$center <- my.fcc$year + my.fcc$period/2

  fcc.breaks$period <- c(fcc.breaks$year[2:nrow(fcc.breaks)] - fcc.breaks$year[1:(nrow(f
  fcc.breaks$center <- fcc.breaks$year + fcc.breaks$period/2
```

```

if(zone=="TGO") {
  title    <- "Togo: Changements bruts et nets des surfaces forestières 2003 - 2018"
  ylim     <- c(-35000, 20000)
  ybreaks  <- seq(-35000, 20000, 5000)
  ymbreaks <- seq(-35000, 20000, 2500)
} else {
  title    <- paste0(zone, ": Changements bruts et nets des surfaces forestières 2003")
  ylim     <- c(-15000, 10000)
  ybreaks  <- seq(-15000, 10000, 5000)
  ymbreaks <- seq(-15000, 10000, 2500)
}

if(!is.null(filename)) pdf(filename)

fcc.breaks$netdefor.ha.yr <- fcc.breaks$netch.ha.yr
fcc.breaks$netdefor.ha.yr[fcc.breaks$netdefor.ha.yr > 0] <- 0
fcc.breaks$netregen.ha.yr <- fcc.breaks$netch.ha.yr
fcc.breaks$netregen.ha.yr[fcc.breaks$netregen.ha.yr < 0] <- 0

print(
  fcc.breaks[-nrow(fcc.breaks),] %>% gather(variable, value, defor.ha.yr, regen.ha.yr,
  ggplot(aes(y=value, x = center, width=period-0.7)) +
  geom_bar(data=. %>% filter(variable %in% c("defor.ha.yr", "regen.ha.yr")), aes(fill=variable))
  geom_bar(data=. %>% filter(variable %in% c("netdefor.ha.yr", "netregen.ha.yr")), aes(fill=variable))
  geom_errorbar(data=my.fcc[1:5,], aes(x=center, y=NULL, ymax=defor.ha.yr, ymin=defor.ha.yr),
  geom_hline(aes(yintercept=0)) +
  scale_fill_manual(name = NULL, breaks = c("defor.ha.yr", "netdefor.ha.yr", "netregen.ha.yr"),
  xlab("Année") + ylab("Hectares par année") + ggtitle(title) +
  scale_x_continuous(minor_breaks = NULL, breaks=c(1991, 2000, 2005, 2010, 2015, 2020)),
  scale_y_reverse(breaks=ybreaks, minor_breaks = ymbreaks) + coord_cartesian(ylim=ylim),
  theme_light() + theme(legend.position=c(0,1), legend.box = "horizontal", legend.justification="left"))
)

if(!is.null(filename)) dev.off()
}

# DO THE WORK #####
# Load and rename forest-cover maps -----
maps <- stack(dir(paste0(FCC.CLN.DIR, "/FC", COV.FC, "/TGO"), pattern = "cf.tif", full.names=TRUE),
names(maps) <- paste0("X", substr(names(maps), 5, 8))

# Create forest cover tables for different periods -----
fc.all      <- fc(maps)
write.csv(fc.all, paste0(FCC.RES.DIR, "/TGO/TGO_fc.csv"), row.names=FALSE)

```

xc

```
fc.all <- read.csv(paste0(FCC.RES.DIR, "/TGO/TGO_fc.csv"))

fc.cln <- fc.all
fc.cln <- fc.cln[!fc.cln$year %in% c(1987, 1991, 2000), ]

fcc(fc.cln)
fcc(fc.cln, c(2003, 2018))
fcc(fc.cln, c(2003, 2015, 2017, 2018))

write.xlsx(list("Total" = fcc(fc.cln)), file = paste0(FCC.RES.DIR, "/TGO/TGO_fcc-all-data.xlsx"))
write.xlsx(list("Total" = fcc(fc.cln, c(2003, 2018))), file = paste0(FCC.RES.DIR, "/TGO_fcc-all-data.xlsx"))
write.xlsx(list("Total" = fcc(fc.cln, c(2003, 2015, 2018))), file = paste0(FCC.RES.DIR, "/TGO_fcc-all-data.xlsx"))

plot.fc(fc.cln, "TGO", paste0(FCC.RES.DIR, "/TGO/TGO_fc.pdf"))
plot.fcc(fc = fc.cln, zone = "TGO", breaks=c(2003, 2015, 2018), filename=paste0(FCC.RES.DIR, "/TGO_fcc.pdf"))

registerDoParallel(.env$numCores-1)
foreach(i=1:length(TGO.reg)) %do% {

  region <- TGO.reg[i,]
  # fc.all <- fc(maps, aoi=region)
  # write.csv(fc.all, paste0(RESULTS.DIR, "/tables/", region$NAME_1, "_fc.csv"), row.names=F)

  fc.all <- read.csv(paste0(RESULTS.DIR, "/tables/", region$NAME_1, "_fc.csv"))

  fc.cln <- fc.all[!fc.all$year %in% c(1987, 1991, 2000), ]

  plot.fc(fc.cln, region$NAME_1, paste0(RESULTS.DIR, "/figures/", region$NAME_1, "_fc.pdf"))
  plot.fcc(fc.cln, region$NAME_1, breaks=c(2003, 2015, 2018), paste0(RESULTS.DIR, "/figures/", region$NAME_1, "_fc.pdf"))

  write.xlsx(list("Total" = fcc(fc.cln)), file = paste0(RESULTS.DIR, "/tables/", region$NAME_1, "_fc.xlsx"))
  write.xlsx(list("Total" = fcc(fc.cln, c(2003, 2018))), file = paste0(RESULTS.DIR, "/tables/", region$NAME_1, "_fc.xlsx"))
  write.xlsx(list("Total" = fcc(fc.cln, c(2003, 2015, 2018))), file = paste0(RESULTS.DIR, "/tables/", region$NAME_1, "_fc.xlsx"))

}

# creating GIS layers for forest loss and forest gain per date ----

forest.loss <- raster(maps)
forest.gain <- raster(maps)      # create empty rasters from stack
forest.pote <- raster(maps)

for(i in 1:(nlayers(maps)-1)) {
  print(paste0("Checking deforestation / reforestation in year ", maps.dates[i]))
```

```

forest.loss[is.na(forest.loss) & is.na(forest.gain) & maps[[i]] == 1 & maps[[i+1]] ==
forest.gain[maps[[i]] %in% c(2,3) & maps[[i+1]] == 1] <- maps.dates[i]
forest.pote[maps[[i]] == 3 & maps[[i+1]] == 2] <- maps.dates[i]
}

# special layer for forest gain in areas where loss has been observed before
loss.gain <- raster(maps)
loss.gain[!is.na(forest.loss) & !is.na(forest.gain) & forest.gain > forest.loss] <- forest.loss

forest.2018 <- maps$X2018
forest.2018[forest.2018==2] <- 3

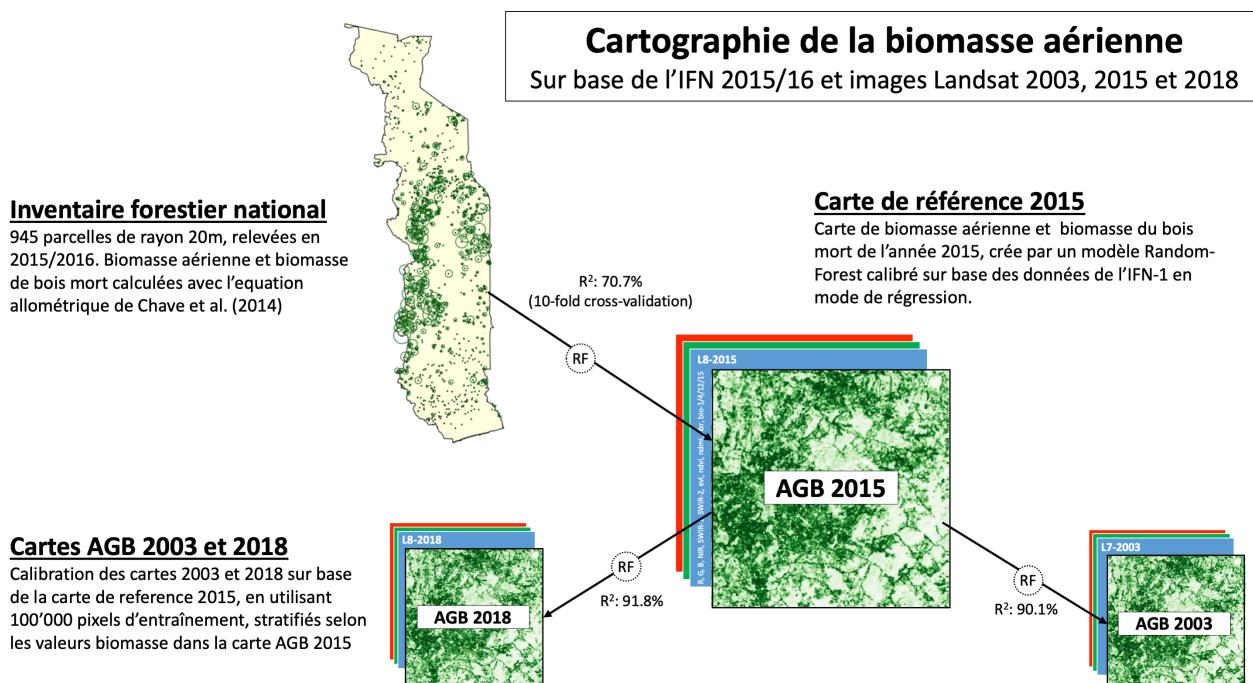
writeRaster(forest.2018, paste0(RESULTS.DIR, "/maps/forest-2018.tif"), datatype="INT")
writeRaster(forest.loss, paste0(RESULTS.DIR, "/maps/forest-loss.tif"), datatype="INT")
writeRaster(forest.gain, paste0(RESULTS.DIR, "/maps/forest-gain.tif"), datatype="INT")
writeRaster(forest.pote, paste0(RESULTS.DIR, "/maps/forest-potential.tif"), datatype="INT")
writeRaster(loss.gain, paste0(RESULTS.DIR, "/maps/loss-gain.tif"), datatype="INT")

```

0.4.2 Analyse biomasse aérienne

Comme pour le couvert forestier, la biomasse aérienne est également cartographiée à la résolution des images Landsat (30 x 30 mètres). Tout d'abord, les données d'inventaire IFN-1 des années 2015/16 sont évaluées pour leur biomasse aérienne. La biomasse aérienne trouvée sur les parcelles d'échantillon IFN-1 sont ensuite utilisées pour calibrer un modèle de régression *RandomForest* sur base des images satellitaires Landsat de l'année 2015 et les données climatiques WorldClim. Ce modèle de biomasse est ensuite appliqué à l'ensemble de l'image Landsat et les données WorldClim pour obtenir une carte complète de la biomasse aérienne pour l'année de référence 2015.

Le schéma ci-dessous montre le processus d'analyse.



La carte de référence de la biomasse aérienne 2015 sert comme base pour déterminer des cartes correspondantes pour les années 2003 (début de la période de référence) et 2018 (fin de la période de référence).

Finalement, les différences entre les cartes de la biomasse 2003 et 2018 servent comme **facteurs d'émission pour les pixels déterminés comme déforestation ou reboisement** par l'analyse des surfaces forestiers au cours de la même période (voir section 0.4.1).

0.4.2.1 Analyse des données IFN

0.4.2.1.1 Description

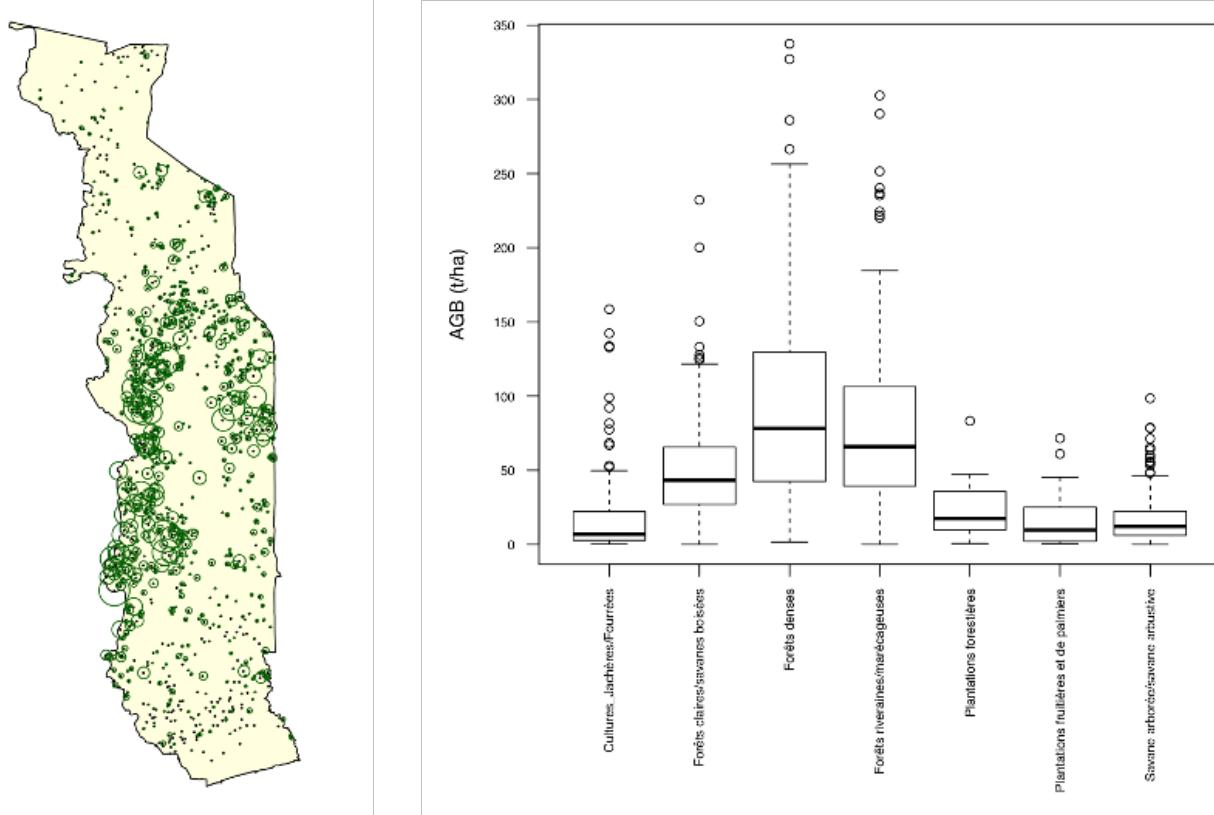
La détermination de la biomasse arborée est basée sur les données de l'inventaire forestier IFN-1 réalisé en 2015/16, en utilisant les données des parcelles (coordonnées du centre de la parcelle et strate de couverture du sol IFN), ainsi que les espèces d'arbres, le diamètre à hauteur poitrine (D à 1.3m) et la hauteur de **tous les arbres avec DHP >= 10 cm enregistrés dans un rayon de 20 mètres autour du centre de la parcelle.**

Toutes les espèces d'arbres sont attribuées avec leur **densité de bois basale ou infradensité, compilée par l'étude de la biomasse de Fonton et al (2018)** en utilisant les bases de données **GlobalWoodDensityDatabase de Zanne et al. (2009)** et **ICRAF Wood Density**. La fonction allométrique de Chave et al. (2014) est utilisé pour estimer la biomasse aérienne des arbres sur base de la densité du bois ρ , du diamètre à hauteur poitrine D et de la hauteur totale H des arbres: $B_{\text{aérienne}} = 0.0673(\rho D^2 H)^{0.976}$. Selon l'étude de Fonton et al. 2018 **la fonction allométrique de Chave et al. (2014) est la fonction la plus approprié pour l'estimation de la biomasse aérienne des arbres au Togo.**

Par la suite, les biomasses des arbres en surface sont additionnées pour toutes les parcelles et converties en tonnes de matière sèche par hectare. Cela est fait séparément pour les arbres vivants et les arbres morts. Sur la base de la biomasse aérienne par hectare, la biomasse racinaire est estimée à l'aide du **rapport racine-tige de Mokany et al (2006) pour les forêts sèches tropicales** ($0,563 \pm 0,086$ pour les forêts avec $B_{\text{aérienne}} = < 20 \text{ t/ha}$ et $0,275 \pm 0,003$ pour les forêts avec $B_{\text{aérienne}} > 20 \text{ t/ha}$).

0.4.2.1.2 Example

Répartition spatiale des **945 placettes d'inventaire de l'IFN-1** (la taille du cercle correspond à la biomasse trouvée sur les placettes) et répartition de la biomasse aérienne dans les différentes strates de l'IFN (données en tonnes de matière sèche par hectare).



Biomasse déterminée à partir des données IFN-1 (en tonnes de matière sèche par hectare) par strate IFN et par compartiment : biomasse aérienne des arbres, bois mort, biomasse racinaire des arbres, et biomasse totale des arbres.

Strate IFN	\$n\$	\$B_{\text{aérienne}}\$		\$B_{\text{morte}}\$		\$B_{\text{racine}}\$	
		\$\mu\$	\$\sigma\$	\$\mu\$	\$\sigma\$	\$\mu\$	\$\sigma\$
Cultures_Jachères/Fourrées	108	20.0	31.7	2.3	8.6	6.6	
Forêts claires/savanes boisées	251	50.1	33.5	2.1	5.2	14.4	
Forêts denses	138	96.8	69.7	3.8	7.5	26.8	
Forêts riveraines/marécageuses	101	84.1	66.1	3.5	5.9	23.3	
Plantations forestières	24	23.3	19.6	0.1	0.1	7.9	
Plantations fruitières et de palmiers	19	18.9	21.0	0.4	1.4	6.0	
Savane arborée/savane arbustive	277	16.4	15.4	0.7	1.6	6.3	
NA	27	0.0	0.0	0.0	0.0	0.0	

0.4.2.1.3 Script R: 03_NRF-MRV/02_AGB/_src/01_compile-AGB.R

```
#####
# 01_compile-AGB.R: Évaluation de la biomasse des données de l'IFN
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables -----
```

```

PLOT.SIZE <- 20^2*pi # Taille de la parcelle en mètres carrés

# Rapports racines-tige forêts tropicales sèches selon Mokany et al. (2006)
# https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2486.2005.001043.x
RSR.Y <- 0.563      # moyenne pour la jeune forêt
RSR.Y.SE <- 0.086    # écart type pour la jeune forêt
RSR.O <- 0.275      # moyenne pour les forêts matures
RSR.O.SE <- 0.003    # écart type pour les forêts matures
RSR.AGB <- 20        # Seuil biomasse aérienne jeunes forêts <-> forêts matures

# Lire et préparer les données IFN-1 =====
plots <- read.xlsx(paste0(DIR.RAW.DAT, "/IFN/IFN-Togo-2015.xlsx"), "placettes")[,1:4]
names(plots) <- c("PlotID", "X", "Y", "LULC")

trees <- read.xlsx("~/Downloads/IFN-Togo-2015.xlsx", "arbres")[,c(1,4:6,8)]
names(trees) <- c("PlotID", "Species", "Status", "DBH", "H")

# Tableau avec les espèces et le nombre d'observations
species <- as.data.frame(table(trees$Species))
names(species) <- c("Species", "Count")

# fusionner avec les densités de bois basales -----
# Densité basale: rapport entre mass anhydre et volume vert (à l'état saturé)
# Source des données: étude biomasse de Fonton et al. 2018
fonton <- read.csv2(paste0(DIR.RAW.DAT, "/IFN/donnees_Tg_Fonton.csv"), encoding="latin1")
fonton <- aggregate(list(D=fonton$wsg), by=list(Species=fonton$NOM), FUN=modal)
species <- merge(species[,c("Species", "Count")],
                  fonton[,c("Species", "D")], by="Species")
species$Source <- "Fonton"

# fusionner avec les arbres dans l'IFN-1
trees <- merge(trees, species, by="Species", all.x=TRUE)

# estimer la biomasse aérienne des arbres -----
# en utilisant la fonction allométrique de Chave et al. (2014)
# https://onlinelibrary.wiley.com/doi/abs/10.1111/gcb.12629
trees$AGB <- 0.0673 * (trees$D * trees$DBH^2 * trees$H)^0.976

# distinguer la biomasse des arbres vivants et celle des arbres morts
trees$AGBm <- trees$AGBv <- trees$AGB      # Dupliquer les valeurs de la biomasse ...
trees$AGBv[trees>Status != "V"] <- 0        # ... et mettre à 0 aux endroits appropriés
trees$AGBm[trees>Status == "V"] <- 0

# biomasse aérienne et bois mort par parcelle (somme des arbres) -----
plots <- merge(plots,
               aggregate(trees[,c("AGBv", "AGBm")],
                         by=list(PlotID=trees$PlotID),
                         # somme dbiomasse es arbres -> à l'héctare -> en tonne

```

```

                FUN=function(x) sum(x) * 10000 / PLOT.SIZE / 1000),
by="PlotID", all.x=TRUE)
# mettre à 0 la biomasse et le bois mort pour les parcelles sans valeurs (NA)
plots$AGBv[is.na(plots$AGBv)] <- 0
plots$AGBm[is.na(plots$AGBm)] <- 0
plots$AGB <- plots$AGBv + plots$AGBm

# estimer la biomasse racinaire par parcelle -----
# avec les facteurs root-shoot de Mokany et al. (2006)

# jeunes forêts
plots$BGB[plots$AGBv <= RSR.AGB] <- plots$AGBv[plots$AGBv <= RSR.AGB] * RSR.Y
# forêts matures
plots$BGB[plots$AGBv > RSR.AGB] <- plots$AGBv[plots$AGBv > RSR.AGB] * RSR.O

# Biomasse totale = biomasse aérienne (vivant et mort) + biomasse racinaire
plots$BM <- plots$AGB + plots$BGB

# Sauvegarder le résultat et production des figures et tableaux =====
# Note: sur Mac utiliser fileEncoding = "macintosh"

write.csv(plots, paste0(DIR.MRV.AGB.REF, "/IFN-plots.csv"),
          row.names = FALSE) #, fileEncoding = "macintosh")

# distribution des biomasses par strate IFN -----
# boxplot biomasse aérienne
pdf(paste0(DIR.MRV.AGB.REF, "/AGB-vs-LULC.pdf"))
par(mar=c(11,5,1,1), cex.axis=0.7)
boxplot(plots$AGBv~plots$LULC, las=2, ylab="AGB (t/ha)", xlab=NULL)
dev.off()

# boxplot bois mort
pdf(paste0(AGB.REF.DIR, "/AGBm-vs-LULC.pdf"))
par(mar=c(11,5,1,1), cex.axis=0.7)
boxplot(plots$AGBm~plots$LULC, las=2, ylab="Bmort (t/ha)", xlab=NULL)
dev.off()

# tableau biomass per LU/LC category (moyenne et écart type)
bm.lulc.tab <- plots %>%
  group_by(LULC) %>%           # grouper par strate
  summarise(n=length(AGB),      # definir colonnes et calcul des valeurs
            AGBv.mean=mean(AGBv), AGBv.sd=sd(AGBv),
            BGB.mean=mean(BGB), BGB.sd=sd(BGB),
            AGBm.mean=mean(AGBm), AGBm.sd=sd(AGBm),
            BM.mean =mean(BM),   BM.sd =sd(BM))
write.csv(bm.lulc.tab, paste0(AGB.REF.DIR, "/AGB_LULC.csv"),
          row.names = FALSE)

```

```

    row.names = FALSE) #, fileEncoding = "macintosh")

# différences biomasse par type de conversion LU/LC -----
dbm.lulc.tab <- bm.lulc.tab %>%
  expand(FROM = nesting(LULC, AGB.mean, AGB.sd, BGB.mean, BGB.sd, BM.mean, BM.sd),
         TO   = nesting(LULC, AGB.mean, AGB.sd, BGB.mean, BGB.sd, BM.mean, BM.sd)) %>%
  mutate(from = FROM$LULC, to = TO$LULC,
         dAGB.mean = TO$AGB.mean - FROM$AGB.mean, dAGB.sd=sqrt(FROM$AGB.sd^2 + TO$AGB.sd^2),
         dBGB.mean = TO$BGB.mean - FROM$BGB.mean, dBGB.sd=sqrt(FROM$BGB.sd^2 + TO$BGB.sd^2),
         dB.Mean = TO$BM.mean - FROM$BM.mean, dB.Msd =sqrt(FROM$BM.sd^2 + TO$BM.sd^2),
         select(c(from, to, dAGB.mean, dAGB.sd, dBGB.mean, dBGB.sd, dB.Mean, dB.Msd))
  write.csv(dbm.lulc.tab, paste0(AGB.REF.DIR, "/AGB_LULC-diff.csv"),
            row.names = FALSE) #, fileEncoding = "macintosh")

```

0.4.2.2 Calibration et prédition

0.4.2.2.1 Description

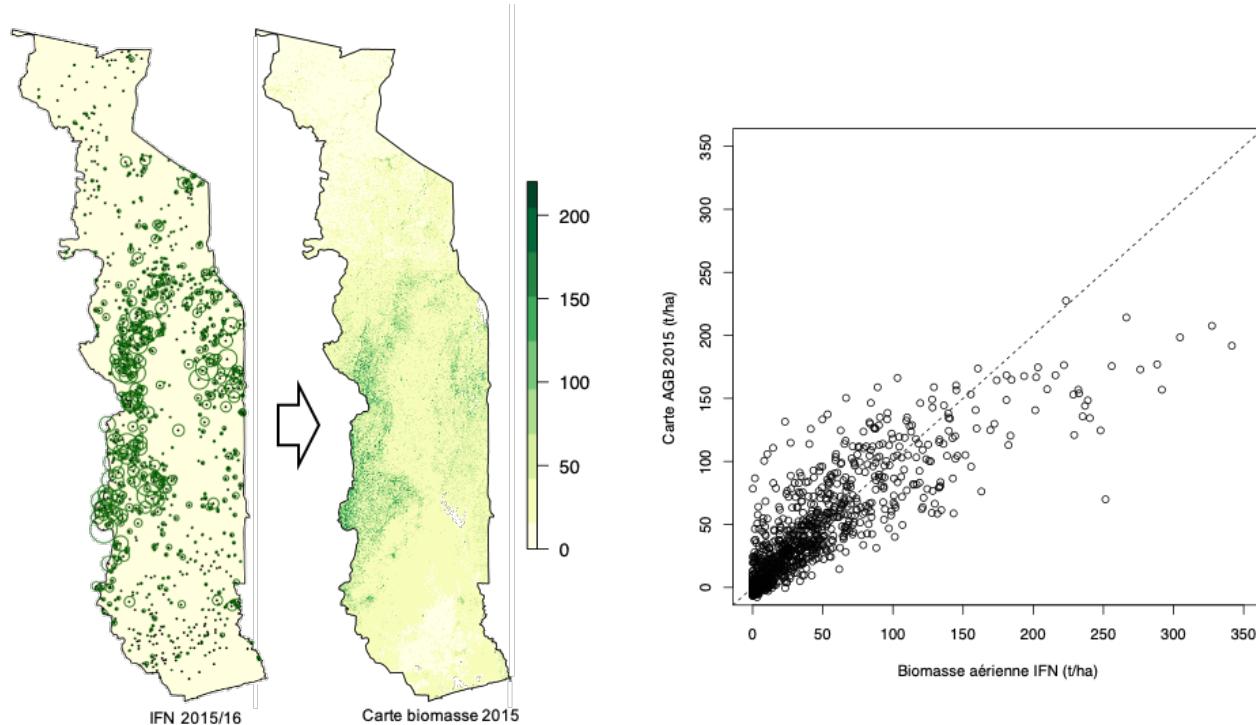
A partir de la **biomasse aérienne déterminée sur les parcelles IFN en 2015/16** (t/ha matière sèche), un modèle de biomasse est calibré sur bases de l'image satellitaire Landsat de 2015 (bandes et indices B, G, R, NIR, SWIR1, SWIR2, nbr, ndmi, ndvi, evi) et les données climatiques Worldclim (BI01, BI04, BI012, BI015). Ce sont les mêmes variables qui ont déjà été utilisées pour la classification forêt/non-forêt.

Dans une première étape, les variables sont extraites, c'est-à-dire que les valeurs des pixels sous les rayons des parcelles sont lues à partir des images Landsat et Worldclim et moyennées en conséquence. Ensuite, les variable sont mis en correspondance avec les biomasses aériennes à travers un **modèle de régression RandomForest**. Ce modèle est utilisé pour créer une carte de la biomasse aérienne pour l'année 2015.

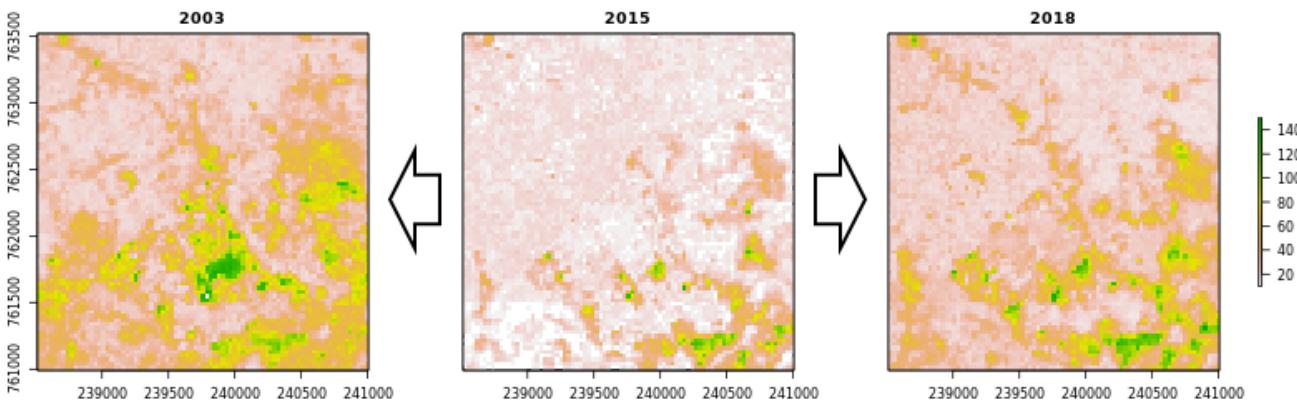
En supposant que la biomasse sur la plupart des pixels reste constante, la **carte de la biomasse aérienne 2015 est utilisée comme base pour calibrer des cartes pour les années 2003 et 2018** (début et fin de la période de référence) avec la même méthode: 0,1% des pixels de la carte de la biomasse 2015 (environ 100 000 pixels) sont utilisés comme pixels d'entraînement pour la calibration d'un modèle RandomForest qui est ensuite utilisé pour la production de la carte.

0.4.2.2.2 Example

Répartition spatiale des **945 placettes d'inventaire de l'IFN-1** (la taille du cercle correspond à la biomasse trouvée sur les placettes) et carte de la biomasse aérienne résultant. Le diagramme de dispersion montre la correlation entre la biomasse sur la carte et celle trouvé sur les parcelles de l'IFN ($R^2 = 70.7\%$). La carte sous-estime les fortes biomasses (effet de saturation).



Utilisation de la **carte de biomasse 2015** comme référence pour la calibration des cartes **2003 et 2018**. Le détail montre une région au sud de Kpalimé.



0.4.2.2.3 Script R: 03_NRF-MRV/02_AGB/_src/02_create-AGB-maps.R

```
#####
# 02_create-AGB-maps.R: Modélisation et création des cartes de biomasse
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables =====
AGB.STRATA      <- 10      # Nombre de strates de biomasse pour l'échantillonnage
N.PIXELS        <- NA      # Nombre de pixels non-NA (sera déterminé plus tard)
SAMPLE.RATIO    <- 0.001   # Rapport des pixels non-NA à échantillonner
```

```

CAL.RATIO      <- 1           # Rapport des pixels à échantillonner pour la calibration

# Bandes à utiliser pour la modélisation forêt vs. non-forêt
PREDICTORS   <- c("B", "G", "R", "NIR", "SWIR1", "SWIR2",
                  "nbr", "ndmi", "ndvi", "evi",
                  "BIO1", "BIO4", "BIO12", "BIO15")

# Répertoires
LANDSAT.DIR <- DIR.SST.DAT.LST
WORLDCLIM.DIR <- DIR.SST.DAT.WC2
REF.DIR <- DIR.MRV.AGB.REF

# Définitions des fonctions =====
# Charger un image Landsat -----
# 
# @param filename Chemin du fichier landsat
# 
# @return          Image Landsat avec bandes nommées
# 

load.image <- function(filename) {
  image <- brick(paste0(LANDSAT.DIR, filename))
  names(image) <- SST.LSBANDS
  return(image)
}

# Prédiction d'une carte biomasse -----
# 
# @param image      Image Landsat à classifier
# @param filename   Nom de fichier pour la sauvegarde de la carte
# @param bioclim    Raster des variables bioclimatiques à utiliser
# @param train.dat  Tableau des données d'entraînement
# @param ref.map    Carte de référence
# @param n.ref.map  Nombre de points à échantillonner
# @param cal.map    Carte de calibration (autre chemin WRS)
# @param n.cal.map  Nombre de points à échantillonner
# @param mask       Masque à utiliser pour ref.map et cal.map
# @param preds      Variables à utiliser pour le modèle
# @param type       Modèle de classification ou de regression
# @param crossval   Faire validation croisée (3 * 10-fold)
# @param bias.corr  Faire une correction linéaire du bias
# @param n.cores    Nombre de processeurs à utiliser pour prédiction
# 
# @return            List avec les éléments
#                   - model RandomForest,
#                   - model linéaire de correction du bias
#                   - carte biomasse
#
#
# 
```

```

agb.map <- function(image, filename, bioclim=NULL, train.dat=NULL,
                     ref.map=NULL, n.ref.map=NULL,
                     cal.map=NULL, n.cal.map=NULL,
                     mask=NULL, preds=NULL,
                     crossval=FALSE, bias.corr=TRUE, ncores=8) {

  # Ouvrir le fichier journal
  name <- sub("[.]tif$", "", filename)
  txtfile <- paste0(sub("[.]tif$", "", filename), ".txt")
  cat("-- Biomass map: ", basename(filename), "/", date(), " --\n", file=txtfile)

  # Tirer des données d'entraînement d'une carte de référence -----
  if(!is.null(ref.map)) {
    cat(paste0(" -Masking / buffering reference map ... \n"))
    # ... couper/masquer avec l'image
    ref.map <- mask(crop(ref.map, image[[1]]), crop(image[[1]], ref.map))
    # ... et masque additionnelle (si disponible)
    if(!is.null(mask)) ref.map <- mask(ref.map, mask)
    # Découper la carte de calibration (si disponible)
    if(!is.null(cal.map)) {
      tmp <- extend(crop(cal.map, ref.map), ref.map)
      ref.map <- mask(ref.map, tmp, inverse=TRUE)
    }
    cat(" ")
    # Tirer des points d'échantillon (stratifié selon classe biomasse) ...
    cat(paste0(" -Sampling map (n=", AGB.STRATA, "*",
               round(n.ref.map/AGB.STRATA), ")"))
    ref.pts <- sampleStratified(cut(ref.map, AGB.STRATA),
                                 round(n.ref.map/AGB.STRATA), sp=TRUE)[,-1]
    names(ref.pts) <- "AGB"
    # ... extraire les valeurs spectrales et bioclimatique correspondantes ...
    cat("extracting values ... \n")
    ref.dat <- cbind(AGB=raster::extract(ref.map, ref.pts, df=TRUE)[,-1],
                     raster::extract(image, ref.pts, df=TRUE)[,-1],
                     raster::extract(bioclim, ref.pts, df=TRUE)[,-1])

    cat("Ref-map points: ", nrow(ref.dat), "/", ref.map@file@name, file=txtfile, append=TRUE)
    # ... et ajouter aux données d'entraînement (si disponible)
    if(is.null(train.dat)) {
      train.dat <- ref.dat
    } else {
      train.dat <- rbind(train.dat, ref.dat)
    }
  }

  # Ajouter des points d'une carte de calibration -----
  if(!is.null(cal.map)) {
    cat(paste0(" -Masking calibration map ... \n"))
    # ... couper/masquer avec l'image
    cal.map <- mask(crop(cal.map, image[[1]]), crop(image[[1]], cal.map))
  }
}

```

```

# ... et masque additionnelle (si disponible)
if(!is.null(mask)) cal.map <- mask(cal.map, mask)
# Tirer des points d'échantillon (stratifié selon classe biomasse) ...
cat(paste0(" -Sampling map (n=", AGB.STRATA, "*",
           round(n.cal.map/AGB.STRATA), ")"))
cal.pts <- sampleStratified(cut(cal.map, AGB.STRATA),
                           round(n.cal.map/AGB.STRATA), sp=TRUE) [, -1]
names(cal.pts) <- "AGB"
# ... extraire les valeurs spectrales et bioclimatique correspondantes ...
cat("extracting values ... \n")
cal.dat <- cbind(AGB=raster::extract(cal.map, cal.pts, df=TRUE) [, -1],
                  raster::extract(image, cal.pts, df=TRUE) [, -1],
                  raster::extract(bioclim, cal.pts, df=TRUE) [, -1])
# ... et ajouter aux points d'entraînement
if(is.null(train.dat)) {
  train.dat <- cal.dat
} else {
  train.dat <- rbind(train.dat, cal.dat)
}
cat("Cal-map points: ", nrow(cal.dat),
    "from", cal.map@file@name, file=txtfile, append=TRUE)
}

# Nombre total des points d'entraînement
cat("Total points: ", nrow(train.dat), "\n", file=txtfile, append=TRUE)

# Calibration du modèle Random Forest -----
# Utiliser toutes les variables si non-spécifiées dans les paramètres
if(is.null(preds)) {
  preds <- names(image)
  if(!is.null(bioclim)) preds <- c(preds, names(bioclim))
}

# calibrer RandomForest (mode de régression)
cat(" -Calibrating RandomForest ... ")
sink(txtfile, append=TRUE)
# Utiliser caret::train pour validation croisée (a besoin de beaucoup de temps)
if(crossval) {
  map.model.cv <- train(y = train.dat[, 1],
                         x = train.dat[, preds],
                         method = "rf", # RandomForest
                         importance = TRUE,
                         trControl = trainControl(
                           method = "repeatedcv",
                           number = 10, # k-fold
                           repeats = 3)) # répétitions

  print(map.model.cv)
  map.model <- map.model.cv$finalModel
  print(map.model)
  cat("\n")
  print(varImp(map.model, scale=FALSE))
# autrement randomForest directement
}

```

```

} else {
  map.model <- randomForest(y=train.dat[,1], x=train.dat[,preds],
                             importance=TRUE) # , do.trace=100)
  print(map.model)
  cat("\n")
  print(varImp(map.model))
}
sink()
# Mesures des erreurs
cat("R2:", round(map.model$rsq[500], 2),
    "RMSE:", round(sqrt(map.model$mse[500]), 2), "\n")
# Model linéaire de correction du bias
if(bias.corr) {
  bc <- lm(train.dat$AGB ~ map.model$predicted)
  sink(paste0(name, "_bc.txt"), split=TRUE)
  print(summary(bc))
  sink()
  save(bc, file=paste0(name, "_bc.RData"))
} else {
  bc <- NULL
}
# Diagramme de dispersion prédition vs. observation
pdf(paste0(name, ".pdf"))
plot(train.dat$AGB ~ map.model$predicted,
      ylab="AGB (tDM/ha)", xlab="Predicted AGB (tDM/ha)")
abline(0,1, lty=2)
if(bias.corr) {
  abline(bc, lty=3, col="red")
}
dev.off()

# Classification de la carte biomasse -----
dir.create(dirname(filename), recursive=TRUE, showWarnings=FALSE)
cat(" -Creating map ... ")
# empiler les couches Landsat et bioclim
if(!is.null(bioclim)) image <- stack(image, crop(bioclim, image))
# classifier l'image sur différents processeurs en parallèle
beginCluster(n=ncores)
map <- clusterR(image, predict, args=list(model=map.model))
endCluster()
# appliquer la correction du bias (si disponible)
if(bias.corr) {
  cat("Applying linear bias correction ...")
  map <- calc(map, fun=function(x){bc$coefficients[1] + bc$coefficients[2]*x})
}
# sauvegarder la carte
cat("writing map ... ")
map <- writeRaster(map, filename=filename,
                    format="GTiff", datatype="INT2U", overwrite=TRUE)
cat("done\n")

```

```

cat("-- Done: ", basename(filename), "/", date(), " --\n", file=txtfile, append=TRUE)

invisible(list(
  "rf.model" = map.model,
  "bc.model" = bc,
  "map"      = map
))
}

# COMMENCER LE TRAITEMENT #####
# Carte de référence 2015 =====

# Charger images 2015 -----
ref.p192 <- brick(paste0(IMAGES.DIR, "/p192/p192_2015_m.tif"))
ref.p193 <- brick(paste0(IMAGES.DIR, "/p193/p193_2015_m.tif"))
ref.p194 <- brick(paste0(IMAGES.DIR, "/p194/p194_2015_m.tif"))
names(ref.p192) <- names(ref.p193) <- names(ref.p194) <- BANDS
ref.images <- list(p192=ref.p192, p193=ref.p193, p194=ref.p194)

# Déterminer le nombre de pixels non-NA
N.PIXELS <- list(p192 = ncell(ref.p192[["B"]]) - summary(ref.p192)[["NA's", "B"]],
                  p193 = ncell(ref.p193[["B"]]) - summary(ref.p193)[["NA's", "B"]],
                  p194 = ncell(ref.p194[["B"]]) - summary(ref.p194)[["NA's", "B"]])

# Charger variables bioclim
bioclim.p192 <- brick(paste0(IMAGES.DIR, "/p192/p192_bioclim.tif"))
bioclim.p193 <- brick(paste0(IMAGES.DIR, "/p193/p193_bioclim.tif"))
bioclim.p194 <- brick(paste0(IMAGES.DIR, "/p194/p194_bioclim.tif"))
names(bioclim.p192) <- names(bioclim.p193) <- names(bioclim.p194) <- BIOCLIM
bioclim <- list(p192=bioclim.p192, p193=bioclim.p193, p194=bioclim.p194)

# Charger les données d'inventaire -----
plots      <- read.csv(paste0(REF.DIR, "/IFN-plots.csv")) # , fileEncoding="macintosh")

# Convertir en points spatiales en utilisant les coordonnées
coordinates(plots) <- ~X+Y
proj4string(plots) <- utm.31

# Figure de distribution spatiale des points
pdf(paste0(REF.DIR, "/IFN-plots_location.pdf"),
     width=3.5, height=7)
par(mar=c(1,1,1,1))
plot(spTransform(TG0, utm.31), col="lightyellow")
plot(plots, add=TRUE, col="black", pch=16, cex=0.3)

```

```

plot(plots, add=TRUE, col="darkgreen", pch=1, cex=plots$AGB/100)
dev.off()

# Convertir les polygones des parcelles en points spatiaux (centroïdes)
plots.poly <- SpatialPolygonsDataFrame(gBuffer(plots, byid=TRUE, width=20), plots@data)

# Extraire les valeurs spectrales pour les parcelles IFN -----
registerDoParallel(CORES)
# ... pour chaque image Landsat (chemin WRS) ...
x <- foreach(i=1:length(ref.images), .combine=cbind) %:%
  # ... et pour chaque couche spectrale (bandes et indices) ...
  foreach(j=1:nlayers(ref.images[[i]]), .combine=cbind) %dopar% {
    # ... calculer la valeur moyenne pondérée au dessous de la parcelle IFN
    raster::extract(ref.images[[i]][[j]], plots.poly, weights=TRUE, fun=mean, df=TRUE)
  }

# Extraire les valeurs bioclimatiques pour les parcelles IFN (en parallèle)
x2 <- foreach(i=1:length(bioclim), .combine=cbind) %:%
  foreach(j=1:nlayers(bioclim[[i]]), .combine=cbind) %dopar% {
    raster::extract(bioclim[[i]][[j]], plots, df=TRUE)[,2]
  }

# Séparer les données d'entraînement selon les différents chemins WRS
# (13 variables spectrales et 19 variables bioclim pour chaque chemin WRS)
dat.p192 <- na.omit(as.data.frame(cbind(plots$AGB, x[,1:13], x2[,1:19])))
dat.p193 <- na.omit(as.data.frame(cbind(plots$AGB, x[,14:26], x2[,20:38])))
dat.p194 <- na.omit(as.data.frame(cbind(plots$AGB, x[,27:39], x2[,39:57])))
names(dat.p192) <- names(dat.p193) <- names(dat.p194) <- c("AGB", SSTS.LSBANDS, SSTS.BIO)
train.data <- list(p192=dat.p192, p193=dat.p193, p194=dat.p194)

# Sélection des variables explicatives -----
# TODO: inclure la carte de régénération comme prédicteur !

agb.varsel <- rfe(y = dat.p193[,1],           # biomasse (1ère colonne)
                     x = dat.p193[,-1],        # prédicteurs (reste)
                     sizes = c(4, 6, 8, 10),
                     rfeControl = rfeControl(
                       functions = rfFuncs,      # utiliser RandomForest
                       method   = "repeatedcv",   # validation croisée
                       number   = 10,            # 10-fold
                       repeats  = 3))           # 3 répétitions

sink(paste0(REF.DIR, "/p193_2015_AGB_varsel-fin.txt"), split=TRUE)
predictors(agb.varsel)
print(agb.varsel)
sink()

pdf(paste0(REF.DIR, "/p193_2015_AGB_varsel-fin.pdf"))

```

civ

```
plot(var.sel, type=c("g", "o"))
dev.off()

# Cartes de référence biomasse aérienne 2015 ----

# Chemin WRS p193
set.RSEED(RSEED)
p193.2015.agb <- agb.map(image      = load.image("/p193/p193_2015_m.tif"),
                           bioclim     = bioclim[["p193"]],
                           filename   = paste0(REF.DIR, "/p193_2015_AGB_R.tif"),
                           train.dat  = train.data[["p193"]],
                           preds       = PREDICTORS,
                           crossval   = TRUE,
                           bias.corr  = FALSE,
                           ncores     = 32)

# Chemins WRS p192 and p194, utilisant p193 pour calibration
set.RSEED(RSEED)
p192.2015.agb <- agb.map(image      = load.image("/p192/p192_2015_m.tif"),
                           bioclim     = bioclim[["p192"]],
                           filename   = paste0(REF.DIR, "/p192_2015_AGB_R.tif"),
                           train.dat  = train.data[["p192"]],
                           # calibration avec carte p193 ...
                           cal.map    = raster(paste0(REF.DIR, "/p193_2015_AGB_R.tif")),
                           # ... avec au moins 200 points
                           n.cal.map = max(200, nrow(train.data[["p192"]])*CAL.RATIO),
                           preds       = PREDICTORS,
                           crossval   = FALSE,
                           bias.corr  = FALSE,
                           ncores     = 32,
                           mask        = TGO)

set.RSEED(RSEED)
p194.2015.agb <- agb.map(image      = load.image("/p194/p194_2015_m.tif"),
                           bioclim     = bioclim[["p194"]],
                           filename   = paste0(REF.DIR, "/p194_2015_AGB_R.tif"),
                           train.dat  = train.data[["p194"]],
                           # calibration avec carte p193 ...
                           cal.map    = raster(paste0(REF.DIR, "/p193_2015_AGB_R.tif")),
                           # ... avec au moins 200 points
                           n.cal.map = max(200, nrow(train.data[["p194"]])*CAL.RATIO),
                           preds       = PREDICTORS,
                           crossval   = FALSE,
                           bias.corr  = FALSE,
                           ncores     = 32,
                           mask        = TGO)

# Fusionner les cartes des chemins p192, p193 et p194
agb.2015 <- mask(crop(mosaic(raster(paste0(REF.DIR, "/p192_2015_AGB_R.tif")),
                                raster(paste0(REF.DIR, "/p193_2015_AGB_R.tif"))),
```

```

            raster(paste0(REF.DIR, "/p194_2015_AGB_R.tif")),
            fun=mean), # valeur moyenne pour les superpositions
            TGO),
            TGO,
            filename=paste0(REF.DIR, "/TGO_2015_AGB_R.tif"), overwrite=TRUE)

# Sauvgarder un image de la carte biomasse 2015
pdf(paste0(REF.DIR, "/TGO_2015_AGB_R.pdf"),
     width=3.5, height=7)
par(mar=c(1,1,1,1))
plot(agb.2015, axes=FALSE, col=brewer.pal(9, "YlGn"), zlim=c(0,220))
plot(spTransform(TGO, utm.31), add=TRUE)
dev.off()

# Correction de la sous-estimation des valeurs biomasse élevés ----

# Diagramme de dispersion carte biomasse 2015 vs. observation parcelles IFN
plots.poly$AGB.pred <- raster::extract(agb.2015, plots.poly, weights=TRUE, fun=mean, df=)
pdf(paste0(REF.DIR, "/AGB-model_2015.pdf"))
plot(plots.poly$AGB.pred ~ plots.poly$AGB,
      xlab="Biomasse aérienne IFN (t/ha)",
      ylab="Carte AGB 2015 (t/ha)",
      xlim=c(0,350), ylim=c(0,350))
abline(0, 1, lty="dashed")
dev.off()

# Correction pour réduire la sous-estimation des valeurs élevés
cv <- train(AGB ~ AGB.pred,
            data=plots.poly@data[!is.na(plots.poly$AGB.pred),],
            method = "lm",           # régression linéaire
            trControl = trainControl(
              method = "cv",         # validation croisée
              number = 10))          # 10-fold
model <- lm(AGB ~ AGB.pred, data=plots.poly@data[!is.na(plots.poly$AGB.pred),])
sink(paste0(REF.DIR, "/TGO_2015_AGB_Rc.txt"), split=TRUE)
print(cv)
summary(model)
sink()
agb.2015 <- writeRaster(model$coefficients["(Intercept)"] + model$coefficients["AGB.pred"],
                         paste0(REF.DIR, "/TGO_2015_AGB_R.tif"), overwrite=TRUE)

# Diagramme de dispersion carte biomasse 2015 corrigé vs. observation parcelles IFN
agb.pred.c <- raster::extract(agb.2015, plots.poly, weights=TRUE, fun=mean, df=TRUE)[,2]
pdf(paste0(REF.DIR, "/AGB-model_2015_c.pdf"))
plot(agb.pred ~ plots.poly$AGB,
      xlab="Biomasse aérienne IFN (t/ha)",
      ylab="Carte AGB 2015 (t/ha)",
      xlim=c(0,350), ylim=c(0,350))
abline(0, 1, lty="dashed")

```

cvi

```
dev.off()
```

```
# Carte biomasse 2003 (sur base de la carte 2015) =====
```

```
# Chemins WRS p193
```

```
set.RSEED(RSEED)
```

```
p193.2003.agb <- agb.map(image      = load.image("/p193/p193_2003_m.tif"),
                           bioclim     = bioclim[["p193"]],
                           filename   = paste0(REF.DIR, "/p193_2003_AGB_R.tif"),
                           train.dat  = NULL,
                           ref.map    = raster(paste0(REF.DIR, "/TGO_2015_AGB_R.tif")),
                           n.ref.map  = SAMPLE.RATIO * N.PIXELS[["p193"]],
                           preds      = PREDICTORS,
                           crossval   = FALSE,
                           bias.corr  = TRUE,
                           n.cores    = 32)
```

```
# Chemins WRS p192 and p194, utilisant p193 pour calibration
```

```
set.RSEED(RSEED)
```

```
p192.2003.agb <- agb.map(image      = load.image("/p192/p192_2003_m.tif"),
                           bioclim     = bioclim[["p192"]],
                           filename   = paste0(REF.DIR, "/p192_2003_AGB_R.tif"),
                           train.dat  = NULL,
                           ref.map    = raster(paste0(REF.DIR, "/TGO_2015_AGB_R.tif")),
                           n.ref.map  = SAMPLE.RATIO * N.PIXELS[["p192"]] / (1 + CAL.RATIO),
                           cal.map    = raster(paste0(REF.DIR, "/p193_2003_AGB_R.tif")),
                           n.cal.map  = SAMPLE.RATIO * N.PIXELS[["p192"]] / (1 + 1/CAL.RATIO),
                           preds      = PREDICTORS,
                           crossval   = FALSE,
                           bias.corr  = FALSE,
                           n.cores    = 32,
                           mask       = TGO)
```

```
set.RSEED(RSEED)
```

```
p194.2003.agb <- agb.map(image      = load.image("/p194/p194_2003_m.tif"),
                           bioclim     = bioclim[["p194"]],
                           filename   = paste0(REF.DIR, "/p194_2003_AGB_R.tif"),
                           train.dat  = NULL,
                           ref.map    = raster(paste0(REF.DIR, "/TGO_2015_AGB_R.tif")),
                           n.ref.map  = SAMPLE.RATIO * N.PIXELS[["p194"]] / (1 + CAL.RATIO),
                           cal.map    = raster(paste0(REF.DIR, "/p193_2003_AGB_R.tif")),
                           n.cal.map  = SAMPLE.RATIO * N.PIXELS[["p194"]] / (1 + 1/CAL.RATIO),
                           preds      = PREDICTORS,
                           crossval   = FALSE,
                           bias.corr  = FALSE,
                           n.cores    = 32,
                           mask       = TGO)
```

```
# Fusionner les cartes des chemins p192, p193 et p194
```

```

agb.2003 <- mask(crop(mosaic(raster(paste0(REF.DIR, "/p192_2003_AGB_R.tif")),
                                raster(paste0(REF.DIR, "/p193_2003_AGB_R.tif")),
                                raster(paste0(REF.DIR, "/p194_2003_AGB_R.tif")),
                                fun=mean),
                                TGO),
                                TGO,
                                filename=paste0(REF.DIR, "/TGO_2003_AGB_R.tif"), overwrite=TRUE)

# Carte biomasse 2003 (sur base de la carte 2015) =====

# Chemins WRS p193
set.RSEED(RSEED)
p193.2018.agb <- agb.map(image      = load.image("/p193/p193_2018_m.tif"),
                           bioclim     = bioclim[["p193"]],
                           filename   = paste0(REF.DIR, "/p193_2018_AGB_R.tif"),
                           train.dat  = NULL,
                           ref.map    = raster(paste0(REF.DIR, "/TGO_2015_AGB_R.tif")),
                           n.ref.map = SAMPLE.RATIO * N.PIXELS[["p193"]],
                           preds      = PREDICTORS,
                           crossval   = FALSE,
                           bias.corr  = TRUE,
                           n.cores   = 32)

# Chemins WRS p192 and p194, utilisant p193 pour calibration
set.RSEED(RSEED)
p192.2018.agb <- agb.map(image      = load.image("/p192/p192_2018_m.tif"),
                           bioclim     = bioclim[["p192"]],
                           filename   = paste0(REF.DIR, "/p192_2018_AGB_R.tif"),
                           train.dat  = NULL,
                           ref.map    = raster(paste0(REF.DIR, "/TGO_2015_AGB_R.tif")),
                           n.ref.map = SAMPLE.RATIO * N.PIXELS[["p192"]] / (1 + CAL.RATIO)
                           cal.map    = raster(paste0(REF.DIR, "/p193_2018_AGB_R.tif")),
                           n.cal.map = SAMPLE.RATIO * N.PIXELS[["p192"]] / (1 + 1/CAL.RATIO)
                           preds      = PREDICTORS,
                           crossval   = FALSE,
                           bias.corr  = FALSE,
                           n.cores   = 32,
                           mask       = TGO)

set.RSEED(RSEED)
p194.2018.agb <- agb.map(image      = load.image("/p194/p194_2018_m.tif"),
                           bioclim     = bioclim[["p194"]],
                           filename   = paste0(REF.DIR, "/p194_2018_AGB_R.tif"),
                           train.dat  = NULL,
                           ref.map    = raster(paste0(REF.DIR, "/TGO_2015_AGB_R.tif")),
                           n.ref.map = SAMPLE.RATIO * N.PIXELS[["p194"]] / (1 + CAL.RATIO)
                           cal.map    = raster(paste0(REF.DIR, "/p193_2018_AGB_R.tif")),
                           n.cal.map = SAMPLE.RATIO * N.PIXELS[["p194"]] / (1 + 1/CAL.RATIO)
                           preds      = PREDICTORS,

```

```

            crossval = FALSE,
            bias.corr = FALSE,
            ncores    = 32,
            mask      = TGO)

# Fusionner les cartes des chemins p192, p193 et p194
agb.2018 <- mask(crop(mosaic(raster(paste0(REF.DIR, "/p192_2018_AGB_R.tif")),
                                raster(paste0(REF.DIR, "/p193_2018_AGB_R.tif")),
                                raster(paste0(REF.DIR, "/p194_2018_AGB_R.tif")),
                                fun=mean),
                                TGO),
                                TGO,
                                filename=paste0(REF.DIR, "/TGO_2018_AGB_R.tif"), overwrite=TRUE)

```

0.4.3 Analyse émissions/séquestrations

0.4.3.0.1 Description

Le calcul des émissions de CO₂ dues à la déforestation et à la séquestration par le reboisement au cours de la période de référence 2003–2018 est basé sur les cartes forêt/non-forêt nettoyées de 2003 et 2018, ainsi que sur les cartes de la biomasse aérienne des années correspondantes.

Dans une première étape, la biomasse racinaire est calculée sur la base des cartes de la biomasse aérienne, en utilisant les rapports racine-tige de Mokany et al (2006) pour les forêts sèches tropicales.

Le calcul des émissions de CO₂, et fait sur base des cartes de biomasse (aérienne et racinaire) de l'année 2003, en considérant que les pixels qui ont été classés comme forêt en 2003 et comme non-forêt en 2018. La biomasse restant après la déforestation est pris en compte par la soustraction de la biomasse moyenne des pixels non-forêt en 2018. Pour les pixels où une différence négative en résulte (augmentation de la biomasse malgré la déforestation), celle-ci est fixée à 0. La conversion de la biomasse en CO₂ se fait au moyen d'une teneur en carbone de la biomasse de 0,47 (GIEC, 2006) et du rapport des poids moléculaires CO₂/C de 44/12.

La même procédure est utilisée pour calculer la séquestration due au reboisement entre 2003 et 2018. La biomasse des pixels reboisés de 2003 est soustraite de celle de 2018. Lorsque la différence est négative (diminution de la biomasse malgré le reboisement), elle est fixée à 0.

0.4.3.0.2 Example

Les résultats sont évalués pour l'ensemble du territoire du Togo et les différents régions: La superficie annuelle déboisée (ha/a), la perte moyenne de biomasse et de CO₂ due à la déforestation par hectare déboisé (tCO₂/ha) et la perte annuelle totale sur la zone analysée (tCO₂/a). L'évaluation de la croissance de la biomasse et de la séquestration du CO₂ due au reboisement est effectuée de manière analogue.

Région	Déforestation			Reboisement			Total	
	ha/a	tCO ₂ /ha	tCO ₂ /a	ha/a	tCO ₂ /ha	tCO ₂ /a	ha/a	tCO ₂ /a
TGO	-15'060	-62.7	-944'475	9'824	21.8	213'885	-5'236	-730'590
Centre	-4'683	-95.2	-445'750	2'883	26.5	76'489	-1'799	-369'261
Kara	-1'545	-56.7	-87'631	694	28.7	19'896	-851	-67'735
Maritime	-3'273	-1.6	-5'306	2'467	8.8	21'784	-806	16'479
Plateaux	-5'284	-65.8	-347'859	3'666	22.7	83'157	-1'618	-264'702
Savanes	-275	-52.4	-14'399	114	32.9	3'745	-161	-10'654

0.4.3.0.3 Script R: 03_NRF-MRV/02_AGB/_src/04_analyze-ER.R

```
#####
# 04_analyze-ER.R: Analyse des émissions et séquestrations CO2
# -----
# Bern University of Applied Sciences
# Oliver Gardi, <oliver.gardi@bfh.ch>
# 13 Mai 2020

# Définitions des variables =====

# Rapports racines-tige forêts tropicales sèches selon Mokany et al. (2006)
# https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2486.2005.001043.x
RSR.Y    <- 0.563      # moyenne pour la jeune forêt
RSR.Y.SE <- 0.086      # écart type pour la jeune forêt
RSR.O    <- 0.275      # moyenne pour les forêts matures
RSR.O.SE <- 0.003      # écart type pour les forêts matures
RSR.AGB  <- 20         # Seuil biomasse aérienne jeunes forêts <-> forêts matures

# Teneur en carbone de la biomasse (valeur default IPCC 2006)
# https://www.ipcc-nkgip.iges.or.jp/public/2006gl/pdf/4_Volume4/V4_04_Ch4_Forest_Land...
C.RATIO <- 0.47

# Répertoires des cartes biomasse et des résultats
AGB.REF.DIR <- DIR.MRV.AGB.REF
AGB.RES.DIR <- DIR.MRV.AGB.RES

# Analyse des cartes de biomasse aérienne =====

# Biomasse aérienne par strate IFN (carte RapidEye) ----

agb.2015 <- raster(paste0(AGB.REF.DIR, "/TGO_2015_AGB_R.tif"))
strata   <- raster(paste0(DIR.RAW.DAT, "/RapidEye/TGO_30m.tif"))

zonal(agb.2015, strata, fun="mean")
zonal(agb.2015, strata, fun="sd")
```

cx

```
# Distribution des biomasses aériennes (densité) -----  
  
# Charger cartes biomasse aérienne 2003, 2015 et 2018  
files <- dir(AGB.REF.DIR, pattern="^TGO.*\\.tif$", full.names=TRUE)  
maps <- stack(files)  
names(maps) <- substr(names(maps), 5, 8)  
  
# Sélectionner les années 2003 et 2018  
maps.dat <- as.data.frame(na.omit(maps[[c("X2003", "X2018")]][]))  
  
# production de la figure  
dens.plot <- maps.dat %>% gather(variable, value, X2003, X2018) %>%  
  ggplot(aes(x=value, color=variable, linetype=variable)) +  
  geom_density() +  
  theme_light()  
  
pdf(paste0(AGB.RES.DIR, "/AGB-densities_03-18.pdf"))  
print(dens.plot)  
dev.off()  
  
# Analyze des émissions et séquestrations CO2 2003-2018 =====  
  
# charger cartes de biomasse 2003 et 2018 et calcule de la biomasse racinaire  
agb.2003 <- raster(paste0(AGB.REF.DIR, "/TGO_2003_AGB_R.tif"))  
bgb.2003 <- agb.2003 * RSR.0  
bgb.2003[agb.2003 <= RSR.AGB] <- agb.2003[agb.2003 <= RSR.AGB] * RSR.Y  
  
agb.2018 <- raster(paste0(AGB.REF.DIR, "/TGO_2018_AGB_R.tif"))  
bgb.2018 <- agb.2018 * RSR.0  
bgb.2018[agb.2018 <= RSR.AGB] <- agb.2018[agb.2018 <= RSR.AGB] * RSR.Y  
  
# charger cartes des surfaces forestiers 2003 et 2018  
fc.2003 <- raster(paste0(DIR.MRV.MCF.CLN, "/FC30/TGO/TGO_2003_F30cf.tif"))  
fc.2018 <- raster(paste0(DIR.MRV.MCF.CLN, "/FC30/TGO/TGO_2018_F30cf.tif"))  
  
# Carte des pertes de biomasse -----  
  
# Masque forêt en 2003 et non-forêt en 2018  
defor <- fc.2003 == FOREST & fc.2018 != FOREST  
  
# moyenne biomasse non-forêt (aérienne et racinaire)  
nf.agb <- mean(agb.2018[fc.2018 == 3], na.rm=TRUE)  
nf.bgb <- mean(bgb.2018[fc.2018 == 3], na.rm=TRUE)  
  
# Perte de biomasse due à la déforestation = biomasse 2003 - moyenne biomasse non-forêt  
defor.agb <- mask(agb.2003, defor, maskvalue=0) - nf.agb  
defor.bgb <- mask(bgb.2003, defor, maskvalue=0) - nf.bgb
```

```

# Mettre à 0, là où la perte est négative
defor.agb[defor.agb < 0] <- 0
defor.bgb[defor.bgb < 0] <- 0

# Carte des gains de biomasse -----
# Masque non-forêt en 2003 et forêt en 2018
regen <- fc.2003 != FOREST & fc.2018 == FOREST

# Gain de biomasse due à la régénération = biomasse 2018 - biomasse 2003
regen.agb.2003 <- mask(agb.2003, regen, maskvalue=0)
regen.bgb.2003 <- mask(bgb.2003, regen, maskvalue=0)
regen.agb.2018 <- mask(agb.2018, regen, maskvalue=0)
regen.bgb.2018 <- mask(bgb.2018, regen, maskvalue=0)
regen.agb <- regen.agb.2018 - regen.agb.2003
regen.bgb <- regen.bgb.2018 - regen.bgb.2003

# Mettre à 0, là où le gain est négative
regen.agb[regen.agb < 0] <- 0
regen.bgb[regen.bgb < 0] <- 0

# Calculer émissions et séquestrations pour une région -----
#
# @param defor      Carte des pixels déforestés
# @param defor.agb   Carte des pertes de biomasse aérienne
# @param defor.bgb   Carte des pertes de biomasse racinaire
# @param regen       Carte des pixels régénérés
# @param regen.agb   Carte des gains de biomasse aérienne
# @param regen.bgb   Carte des gains de biomasse racinaire
# @param aoi         Zone d'intérêt (région)
# @param years       Nombre d'années entre les observations
#
# @return            List avec les éléments
#                   - defor.area      Surface déforestée
#                   - defor.area.a    Surface déforestée par année
#                   - defor.agb.ha   Perte de biomasse aérienne par hectare déforestée
#                   - defor.agb.a    Perte de biomasse aérienne par année
#                   - defor.bgb.ha   ... même chose pour la biomasse racinaire ...
#                   - defor.bgb.a    ...
#                   - defor.co2.ha   ... et converti en CO2eq ...
#                   - defor.co2.a    ...
#                   - regen.area     Surface régénérée
#                   - regen.area.a   Surface régénérée par année
#                   - regen.agb.ha   Gain de biomasse aérienne par hectare régénérée
#                   - regen.agb.a    Gain de biomasse aérienne par année
#                   - regen.bgb.ha   ... même chose pour la biomasse racinaire ...
#                   - regen.bgb.a    ...

```

```

#           - regen.co2.ha ... et converti en CO2eq ...
#           - regen.co2.a
#
#
emissions <- function(defor, defor.agb, defor.bgb,
                       regen, regen.agb, regen.bgb, aoi=NULL, years=15) {

  # couper la zone d'intérêt
  if(!is.null(aoi)) {
    defor      <- mask(crop(defor, aoi), aoi)
    defor.agb <- mask(crop(defor.agb, aoi), aoi)
    defor.bgb <- mask(crop(defor.bgb, aoi), aoi)
    regen     <- mask(crop(regen, aoi), aoi)
    regen.agb <- mask(crop(regen.agb, aoi), aoi)
    regen.bgb <- mask(crop(regen.bgb, aoi), aoi)
  }

  # convertir biomasse aérienne et racinaire en CO2eq
  defor.co2   <- (defor.agb + defor.bgb) * C.RATIO * 44/12
  # déterminer la surface (taille d'un pixel Landsat = 30mx30m)
  defor.area  <- sum(defor[], na.rm=TRUE) * 30^2 / 10000
  defor.area.a <- defor.area / years
  # moyenne des pertes (par hectare)
  defor.agb.ha <- mean(defor.agb[], na.rm=TRUE)
  defor.bgb.ha <- mean(defor.bgb[], na.rm=TRUE)
  defor.co2.ha <- mean(defor.co2[], na.rm=TRUE)

  # la même chose pour la régénération
  regen.co2   <- (regen.agb + regen.bgb) * C.RATIO * 44/12
  regen.area  <- sum(regen[], na.rm=TRUE) * 30^2 / 10000
  regen.area.a <- regen.area / years
  regen.agb.ha <- mean(regen.agb[], na.rm=TRUE)
  regen.bgb.ha <- mean(regen.bgb[], na.rm=TRUE)
  regen.co2.ha <- mean(regen.co2[], na.rm=TRUE)

  return(list(
    defor.area    = defor.area,
    defor.area.a = defor.area.a,
    defor.agb.ha = defor.agb.ha,
    defor.agb.a   = defor.agb.ha * defor.area.a,
    defor.bgb.ha = defor.bgb.ha,
    defor.bgb.a   = defor.bgb.ha * defor.area.a,
    defor.co2.ha = defor.co2.ha,
    defor.co2.a   = defor.co2.ha * defor.area.a,
    regen.area   = regen.area,
    regen.area.a = regen.area.a,
    regen.agb.ha = regen.agb.ha,
    regen.agb.a   = regen.agb.ha * regen.area.a,
    regen.bgb.ha = regen.bgb.ha,
  ))
}

```

```

    regen.bgb.a = regen.bgb.ha * regen.area.a,
    regen.co2.ha = regen.co2.ha,
    regen.co2.a = regen.co2.ha * regen.area.a
  ))
}

# COMMENCER LE TRAITEMENT #####
# Analyse des pertes de biomasse et CO2 dà cause de la déforestation
# et les gains à cause de la régénération pour Togo et les régions -----
# traitement en parallèle
registerDoParallel(CORES)
res <- foreach(i=0:length(TGO.REG), .combine=rbind) %dopar% {

  if(i==0) {
    # Analyse pour l'ensemble du Togo
    data.frame(reg = "TGO",
               emissions(defor, defor.agb, defor.bgb, regen, regen.agb, regen.bgb))
  } else {
    # Analyse pour une région
    region <- TGO.reg[i,]
    data.frame(reg = region$NAME_1,
               emissions(defor, defor.agb, defor.bgb, regen, regen.agb, regen.bgb,
                          aoi=region))
  }
}

# sauvegarder le tableau des résultats
write.csv(res, paste0(AGB.RES.DIR, "/NERF-Results.csv"), row.names=FALSE)

```