Université de Genève, Faculté des Sciences de la Société

Certificat Complémentaire de Géomatique

Mémoire 2021-2022

Dirigé par

Gregory Giuliani, Senior Lecturer

Impacts des changements climatiques sur les lacs Baringo (Kenya), Suesca (Colombie) et Tonlé Sap (Cambodge)

Monitoring par séries temporelles satellitaires

Par Simon Genoud

Résumé

L'utilisation des systèmes d'information géographiques est en plein essor. Ces outils s'avèrent être très utiles lorsqu'il s'agit notamment d'effectuer un suivi sur des données climatiques.

Cette étude propose un suivi par séries temporelle satellitaires sur trois régions différentes du globe, atteinte d'une certaine façon par le dérèglement climatique.

C'est avec l'aide de la plateforme « cloud-based » Google Earth Engine que nous proposerons un processus d'analyse de l'évolution de la surface des eaux des lacs Baringo (Kenya), Suesca (Colombie) et Tonlé Sap (Cambodge).

Mots-clés GOOGLE EARTH ENGINE – MONITORING – SERIE TEMPORELLE – SATELLITE

Table des matières

1. INTRODUCTION

1.1 Problématique

En 2020, plus de 5'000 habitants sont forcés à quitter leur habitation, des écoles et hôpitaux détruits et une péninsule devenue île dont la surface se réduit rapidement a engendré une action de délocalisation des rares girafes de Rotschild à la barge[1]. Le lac Baringo au Kenya, notamment, a subi de fortes montées des eaux durant cette dernière décennie.

En contrepartie, au Cambodge, les sécheresses du lac Tonlé Sap perturbent les zones humides du bassin et anéantissent l'habitat de la biodiversité, notamment des poissons, dont quelques milliers de pêcheurs en dépendent pour y vivre[2].

Parallèlement au lac Tonlé Sap, la Lagune de Sueca en Colombie ne reçoit plus du tout assez de précipitations dont son réservoir en dépend pour exister[3]. A l'image de Baringo et Tonlé Sap, ce réservoir de biodiversité et les habitants qui en dépendent sont mis en péril.

Aujourd'hui, le dérèglement climatique global et ses impacts socio-économiques et environnementaux ne sont plus à présenter. Quelles que soient ses véritables causes, les effets restent alarmants et continuent de s'aggraver.

Ce qu'il m'intéresse d'étudier ici ne sont pas les causes de ces dérèglements mais les effets directs (indépendamment de leur origine) que ces derniers ont sur l'environnement. Plus précisément, nous tenterons d'établir une analyse du changement de surface des eaux de ces trois régions du monde afin d'en obtenir une idée visuelle et graphique concrète.

Dans le cadre du certificat complémentaire en géomatique, c'est avec l'aide des SIG (systèmes d'information géographiques) et de données satellitaires très utiles pour effectuer un suivi de l'évolution des surfaces des eaux que s'inscrit cette étude.

Toutes les opérations jusqu'à la génération des données sont effectuées avec le logiciel Google Earth Engine.

---

[1] https://www.nationalgeographic.fr/animaux/kenya-danciens-ennemis-se-sont-unis-pour-construire-larche-des-girafes, consulté le 05 janvier 2022.

[2] https://www.nationalgeographic.com/science/article/cambodia-tonle-sap-lake-running-dry-taking-flooded-forest-fish, consulté le 15 septembre 2021.

[3] https://lakalle.bluradio.com/noticias/la-laguna-de-suesca-agoniza-en-el-corazon-de-colombia, consulté le 15 septembre 2021.

1.2 But de l'étude

Le but de ce mémoire est d'explorer l'utilisation de GEE (Google Earth Engine) et de mettre en œuvre un processus d'analyse pour évaluer les changements de surface des eaux de ces trois lacs et les impacts potentiels sur les populations environnantes.

La mise en place de ce processus permettra, je l'espère, une meilleure compréhension de la dynamique des surfaces des eaux grâce à l'exploitation des séries temporelles d'images satellites.

Ainsi, cette analyse illustrera deux effets contradictoires du dérèglement climatique, et ce sous le point de vue des trois lacs mentionnés plus haut.

1.3 Organisation du mémoire

Pour une claire compréhension, divisons ce travail en plusieurs étapes :

- Dans un premier temps, j'expliquerai la situation géographique et climatique pour nos régions d'intérêts.
- La deuxième partie sera dédiée à une revue de littérature afin de savoir ce que la science a déjà fait dans ce domaine et de connaître les outils et moyens disponibles.
- La troisième partie se consacrera à l'explication de la méthodologie, des moyens utilisés pour obtenir les résultats nécessaires (cartes, graphiques, application).
- La quatrième partie analysera les premiers résultats obtenus, une présentation générale de l'évolution annuelle de la surface des eaux.
- La cinquième partie analysera les deuxièmes résultats obtenus, l'évolution saisonnière des eaux.
- La sixième partie proposera un modèle d'analyse de suivi annuel applicable pour les années futures.
- La dernière partie présentera le développement d'une application web afin de pouvoir visualiser interactivement les résultats.

2. SITUATIONS

Je pense qu'il est important d'expliquer les situations géographiques et climatiques des lacs dont il sera question du long de ce travail. Je propose alors ci-dessous de brèves descriptions des trois régions différentes et de leurs périodes sèches et humides, ce qui sera utile plus tard pour quantifier les résultats.

2.1 LAC BARINGO

Situé au nord de la vallée du Rift au Kenya, le lac Baringo repose dans un climat aride et semi-aride.

Les principaux moyens de subsistance des habitants de cette région sont l'élevage de bétail dont menacent les épisodes de sécheresse enregistrées (Ochieng et al. 2017).

Le lac est alimenté par plusieurs rivières et n'a pas de débouché apparent si ce n'est l'infiltration dans ses sédiments volcaniques. Il est l'un des deux importants lacs d'eau douce de cette région aride, abrite l'habitat pour environ 500 espèces d'oiseaux différentes et est classé site RAMSAR depuis 2002.

La pêche locale ainsi que l'écotourisme sont économiquement importants pour le développement durable des populations environnantes[4].

Mais durant cette dernière décennie, les importantes précipitations ont fait monter les eaux du lac impactant la civilisation et ses activités aux alentours. Multitudes de foyers, élevages, industries et infrastructures ont été perturbés[5].

Le Kenya, et le comté de Baringo sont caractérisés par deux saisons sèches et deux saisons humides : la petite saison des pluies de novembre à décembre, suivie d'une sècheresse de janvier à mars, pour ensuite donner place à la grande saison des pluies d'avril à juin et terminer avec une période sèche jusqu'à novembre, et ainsi de suite.

[4] https://rsis.ramsar.org/fr/ris/1159?language=fr, consulté le 16 janvier 2022.

[5] https://www.tuko.co.ke/374949-lake-baringo-water-levels-rise-tremendously-submerges-hotels-residences.html, consulté le 16 janvier 2022.

2.2 LAC TONLE SAP

Le lac Tonlé Sap se situe au Cambodge. Il est le plus grand lac d'eau douce d'Asie du Sud-Est. En plein dans la plaine inondable cambodgienne, il est alimenté par plusieurs rivières, et notamment par le fleuve du Mékong qui apporte sédiments nécessaires aux habitats naturels et agricoles qu'offre le lac.

Issu d'un complexe cycle hydrologique, ce riche écosystème profite à la biodiversité et aux populations rurales environnantes de par la pèche et l'irrigation des cultures de riz (Wen & Edward 2021).

Avec l'installation de barrages perturbant le flux des drainages des eaux et les sécheresses (dues au changement climatique) enregistrées ces deux dernières décennies, les habitats des poissons sont perturbés ; impactant directement les pêcheries et la sécurité alimentaire du grand nombre de population y dépendant.

Sous l'influence d'un climat tropical, cette région est divisée en deux saisons : la saison des pluies de juin à novembre et la saison sèche les mois restants. Le niveau du lac atteint donc son maximum en novembre à cause de l'augmentation du débit du Mékong, ce qui d'ailleurs dicte le sens de l'écoulement de la rivière Tonlé Sap liant le lac et le fleuve : le courant change de direction quand le lac s'arrête de gonfler et commence à se drainer au début de la saison sèche.

2.3 LAGUNE DE SUESCA

La lagune de Suesca est un lac situé à une centaine de kilomètres au nord de Bogota en Colombie.

Ce réservoir d'eau douce a subi une grande sécheresse et diminution en 2009 mais celle-ci fut contrée par le phénomène de courant côtier chaud El Niño augmentant les précipitations et le niveau des eaux pour les années suivantes. Cependant, l'érosion et la déforestation ajoutés au manque de précipitations enregistré ces quelques dernières années, mettent la lagune en péril (Castellanos & Francy 2005).

Socio économiquement, les habitants en souffrent car ils ne peuvent plus correctement irriguer leurs cultures et désaltérer leur bétail, ou encore accueillir les touristes attirés par ce qu'offrait le lac au temps où il n'était pas asséché.

Concernant le climat, la région a un régime de pluie bimodal fortement influencé par les vents. Il y'a deux saisons des pluies : entre mars et mai, puis en octobre et novembre.
Janvier, février, juillet et août sont les mois les plus secs.

3. REVUE DE LITTERATURE

Afin de quantifier dans le temps l'évolution des surfaces des eaux, il est très utile d'utiliser les séries temporelles d'images satellites.
M. Broich et al. nous expliquent qu'en utilisant des données de couvertures du sol issues des archives Landsat, et en y appliquant une méthode basée sur un algorithme aléatoire de forêt, il est possible de tracer les variations des eaux de surface et les dynamiques des inondations sur un bassin hydrologique d'une région semi-aride (Broich et al. 2016). Les données sont limitées cependant à 2011, mais la démarche est intéressante et m'inspire pour ce travail.
En effet, les images Landsat sont très pratiques puisqu'elles sont disponibles depuis 1984 à aujourd'hui.
Je détaillerai par la suite les propriétés de ces données, vu que je m'en servirai pour produire les résultats.

Avec cette même source de données, Kiage L. et. al. (2007) nous démontrent qu'il est faisable de développer un travail sur la région du lac Baringo en analysant la dégradation de la couverture du sol du comté de Baringo. Cela signifie que des études sur les changements de niveaux des eaux est applicable.

Pour compléter les données disponibles sur la quantification des niveaux des eaux captées par les gauges in situ, les données satellitaires sont très utiles car elles couvrent la totalité du globe.
En quelques sortes, n'importe quelle région du monde est prête à être analysée, contrairement aux stations hydrologiques qui ne sont pas présentes partout dans le monde.
Dans cette optique, Petrakis R. E. et al. (2020) nous présentent un modèle de la dynamique de l'étendue des eaux de surface (DSWE) résultant en plusieurs cartes thématiques contenant différentes classes en fonction du degré de confiance quant à la présence d'eau, et ce pour le territoire du Cambodge et avec GEE.

C’est en m’inspirant de ces méthodes que je tenterai de développer un processus d’analyse pour les trois lacs présentés plus haut.

4. METHODOLOGIE

La première démarche a été la prise de connaissance et l’apprentissage de Google Earth Engine. Ce logiciel open source basé sur le cloud computing (fourniture de services informatiques via Internet) combine un catalogue de plusieurs pétaoctets d’images satellites et de données géospatiales avec des capacités d’analyse à l’échelle planétaire.
L’apprentissage s’est fait en très grande partie à l’aide des tutoriels du Prof. Wu Q. qui propose un package « geemap » fonctionnant sur le langage de programmation Python (Wu 2020). L’éditeur de code de GEE fonctionne par défaut avec le langage JavaScript, mais j’ai choisi d’utiliser la méthode sous Python car d’autres opérations approfondies y sont possibles, telle que l’utilisation d’une boîte d’outils.
Une fois la connaissance acquise, j’ai pu commencer à construire mes scripts afin de développer un processus d’analyse capable de rendre compte du suivi des lacs, en fonction des données disponibles.

4.1 Données à disposition

Le choix des données utilisées pour cette étude s’est construit au fur-et-à-mesure parallèlement à l’apprentissage du fonctionnement de GEE. Au final, j’ai sélectionné et utilisé trois datasets (collections de données) issues du catalogue GEE :

- « **JRC Yearly Water Classification History, v1.3** »

  Ce jeu de données est construit sur la base de 4'453'989 images à 30m de résolution issues des satellites Landsat 5, 7 et 8 acquises entre 1984 et 2020, dans le cadre du programme européen Copernicus (partenariat entre la Commission et l’Agence Spatiale Européenne) « Global Surface Water ».

  Chaque pixel est individuellement classé en tant que « eau / non-eau » en fonction d’un système de détection dérivant l’index **NDVI** et le modèle **HSV**.

Cette classification de l'occurrence hydrologique annuelle prenant en compte la saisonnalité sera très utile à ma première série de résultats visant à dresser une idée globale du niveau des eaux (Pekel et al. 2016).

- « **USGS Landsat 8 Collection 1 Tier 1 TOA Reflectance** »

  Ce deuxième jeu de données est la première collection du plus récent satellite Landsat lancé en février 2013, dirigé par **l'USGS**. Doté du capteur OLI, les images produites sont multispectrales composées de 9 bandes différentes, pour une résolution de 30m.

  La qualité est supérieure à celle des précédentes versions de Landsat, notamment au niveau des corrections des « Scan Line Errors » qui venaient perturber mes résultats lors de la recherche de méthode (je détaillerai plus bas).

  Avec une récurrence de 16 jours (temps entre la production de deux images pour une même zone), ces données me seront utiles pour quantifier les dynamiques des eaux plus en détail qu'annuellement.

- « **Sentinel-2 MSI: MultiSpectral Instrument, Level-2A** »

  Ce troisième jeu de données produit au sein du programme Copernicus est doté d'une résolution supérieure (10m-60m) et surtout de 13 bandes spectrales dont certaines me permettront de contrer le biais des nuages rendant avec d'autres données la production de résultats compliquée, notamment pour la région de Suesca.

  Le satellite a été lancé en 2015. C'est avec cette collection d'images que je pourrai proposer ma dernière méthode d'analyse pour le suivi futur de l'évolution des eaux.

Grace à ces collections d'images libre d'accès à travers GEE et au développement de mes scripts, je peux produire les résultats suivants et atteindre les objectifs de ce mémoire.
Je vais maintenant présenter et détailler les démarches entreprises.

5. RESULTATS & ANALYSES

Tous les détails sont dans les scripts en annexes à la fin du document.
J'essayerai cependant d'expliquer les étapes importantes pour une bonne compréhension des choix effectués.

5.1 Présentation générale de l'évolution annuelle de la surface des eaux (annexe : SCRIPT 1, p33)

Cette première partie a pour but de dresser l'état général de l'évolution des eaux des trois régions afin de se faire une idée globale de leur dynamique.
Selon les articles cités dans l'introduction, il semble que les changements notables de niveaux occurrent dès 2012 pour le lac Baringo, entre 1993 et 2018 pour le lac Tonlé Sap et depuis 2008 pour la lagune de Suesca.

Commençons donc par produire trois figures résumant les collections d'images pour chaque région.
Il faut d'abord définir les trois régions d'intérêts (roi) afin de cibler nos futurs calculs pour ne prendre en compte que les eaux des lacs en question. En fonction des périodes où les eaux s'étendent au maximum, je dessine autour de chaque lac des polygones.
Il faut filtrer notre collection d'images **'JRC/GSW1_3/YearlyHistory'** et créer grâce à la fonction **.getFilmstripThumbURL()** une bande illustrant chacune des 37 images annuelles de la collection.
Pour un souci esthétique je revisite ces résultats avec le logiciel Adobe Photoshop qui me permet d'arranger cette bande en une image plus compacte et d'ajouter les labels des années :

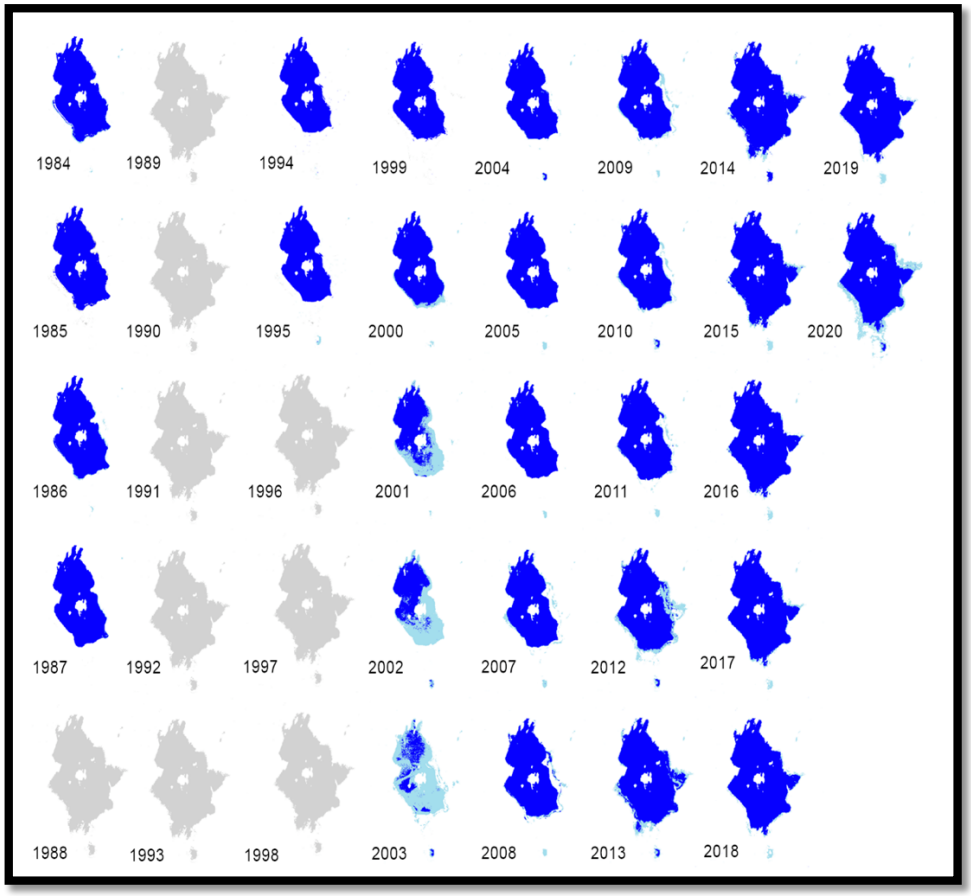

Figure 1: FilmStrip pour la collection JRC du lac Baringo

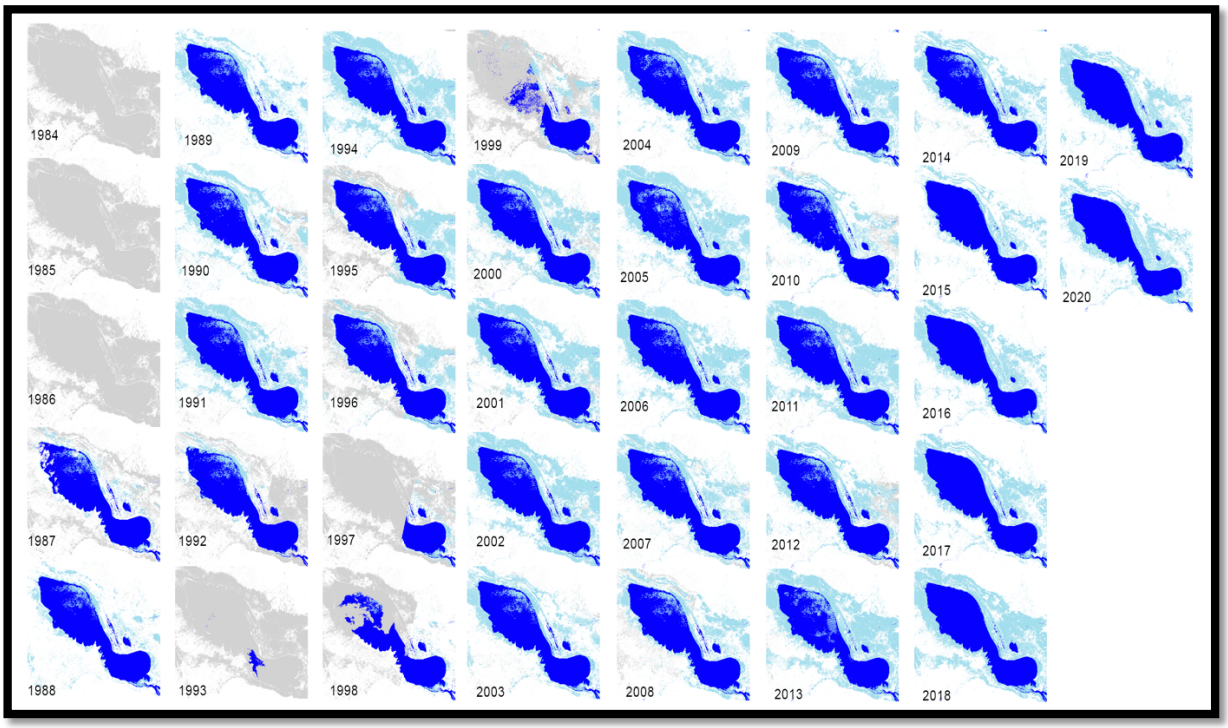

Figure 2: FilmStrip pour la collection JRC du lac Tonlé Sap

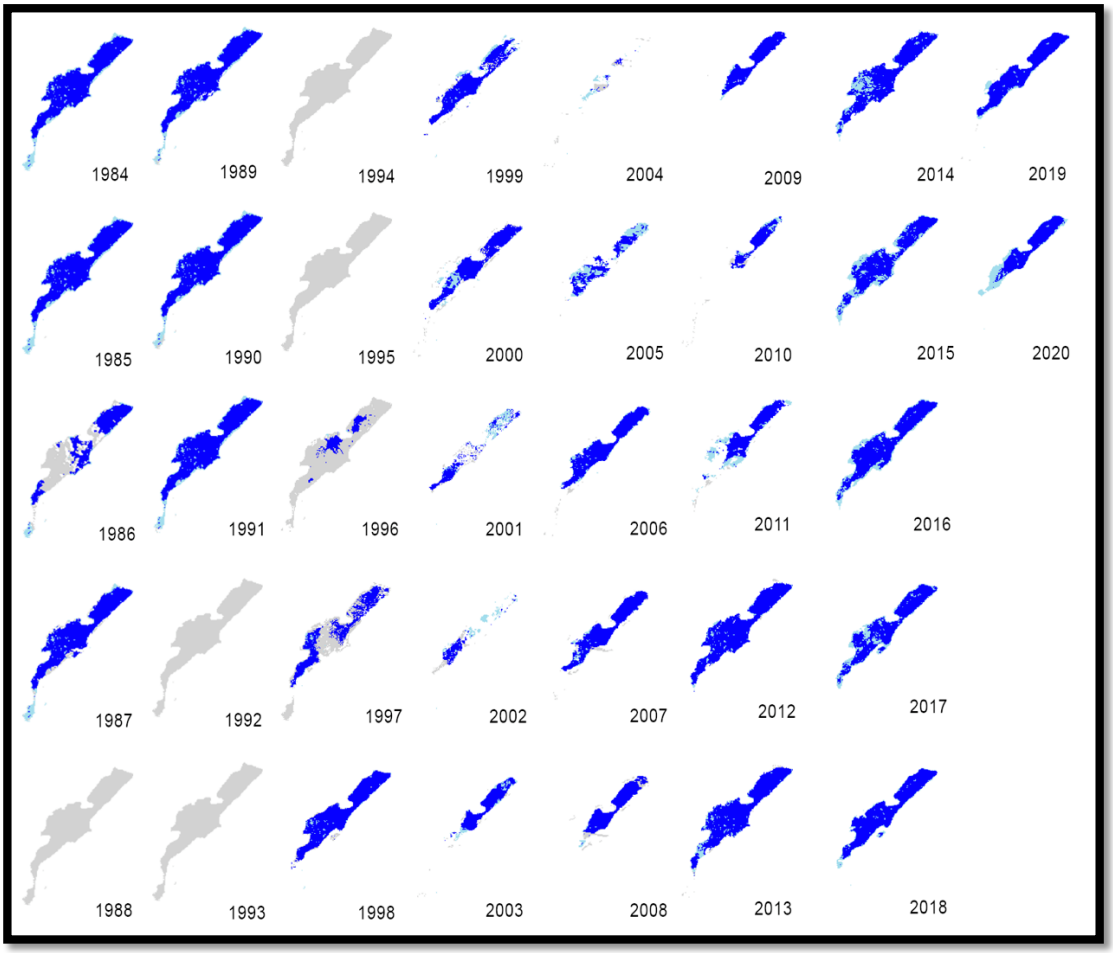

Figure 3: FilmStrip pour la collection JRC du lac Suesca

Le bleu foncé (valeur de 3 de l'unique bande **waterClass** des images de la collection) représente la présence d'eau permanente enregistrée au cours d'une année ; le bleu clair (valeur de 2) représente la présence d'eau saisonnière ; et le gris (valeur de 1) représente aucune eau détectée pour l'année en question.

Visuellement, on s'aperçoit de la claire augmentation des niveaux pour Baringo et d'une dynamique instable pour Tonlé Sap et Suesca.

Globalement, l'incomplétude des données nous permet de pousser l'analyse avec une condition : effectuer un suivi des trois régions à partir de l'année 2000, celle où les données semblent être complètes pour chacun des trois lacs.

Poursuivons alors pour tenter de quantifier cette évolution.

En fonction de la collection définie, je la filtre pour ne garder que les images datant de 2000 et plus.

Ceci est possible grâce aux métadonnées liées à chaque image, un atout des **SIG**.

On peut ensuit définir une fonction qui pour chaque image de la collection, va venir sélectionner les valeurs de la bande unique qui les composent (0, 1, 2 ou 3). Pour chacune de ces bandes on paramètre la fonction pour qu'elle calcule le nombre de pixel présents. Valeur qui sera « **réduite** » en total de pixels, ce qui nous informera sur l'aire totale de chaque classe pour chaque image.

Les résultats nous sont présentés sous forme de liste à laquelle on peut y ajouter les années respectives de chaque image.

J'ai décidé ensuite de remanier les chiffres obtenus à l'aide du logiciel Microsoft Excel afin d'avoir la liberté de produire les graphiques suivants :

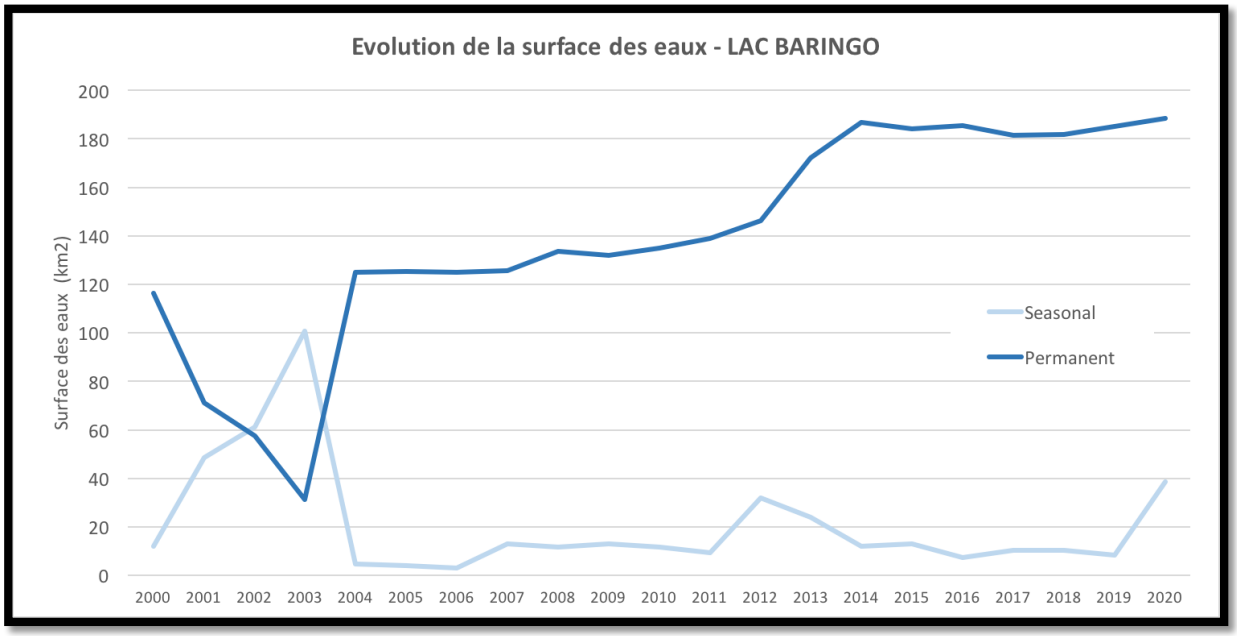

Figure 4: Graphique de l'évolution annuelle du lac Barino (JRC)

Pour le lac Baringo, on observe une augmentation de la surface permanente des eaux depuis 2003.

Notons les deux pics de 2004 et 2013, la surface permanente a gagné alors respectivement +300,1 % et +27,7 %. Si l'on prend en compte la saisonnalité, donc que l'on calcule le total des deux courbes, le premier chiffre s'aplatit mais le deuxième reste notable.

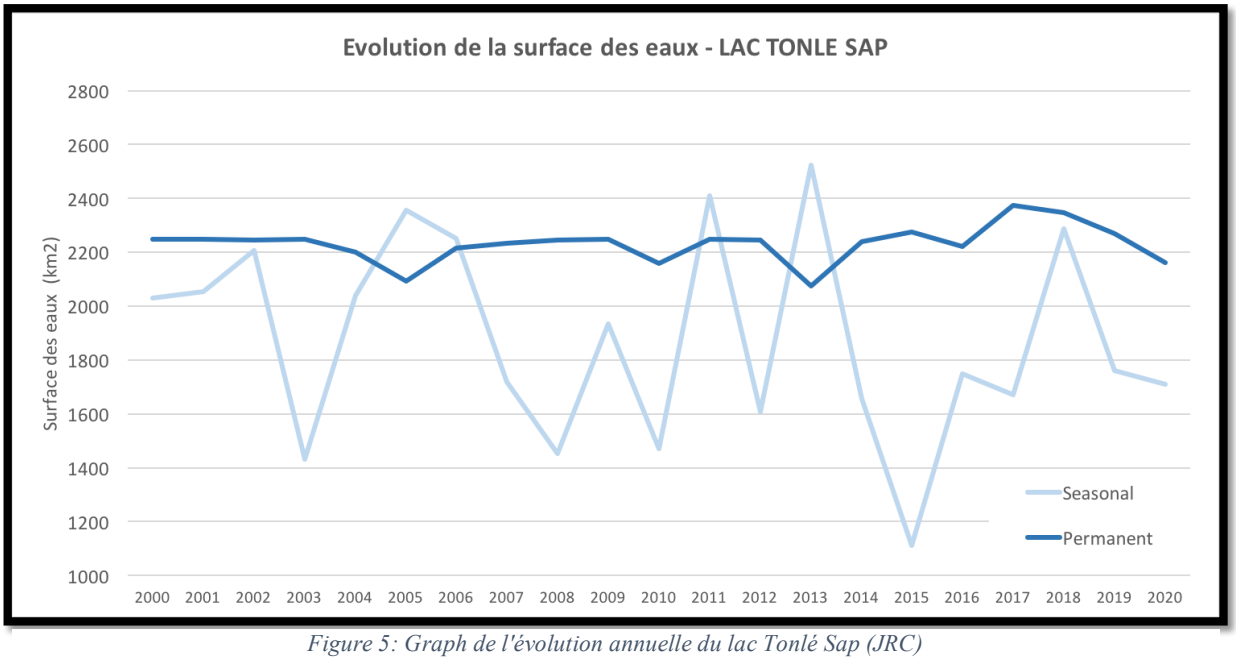

Figure 5: Graph de l'évolution annuelle du lac Tonlé Sap (JRC)

Les eaux du lac Tonlé Sap semblent avoir été plus ou moins stables dans le temps. Comme indiqué plus haut dans la partie situationnelle, ce sont les eaux saisonnières qui fluctuent énormément et cela se confirme à travers ce graphique.

La quantité d'eau enregistrée a été très volatile au cours de ces dernières années avec des bonds passant d'une surface de 2'524,15km2 en 2013 à 1'112,94 km2 en 2015.

On voit là le manque de stabilité que subit cette région annuellement depuis deux décennies.

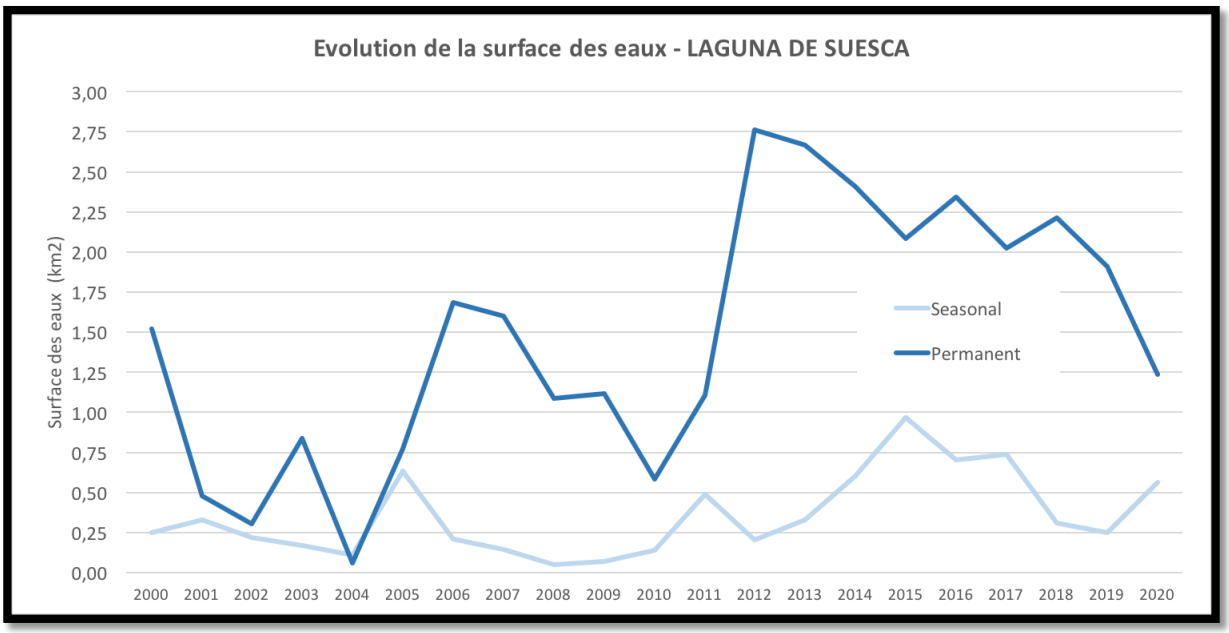

Figure 6: Graph de l'évolution annuelle du lac Suesca (JRC)

Ce troisième graphique témoigne de la sécheresse discutée plus haut dans la région de Suesca. On observe aussi les effets du phénomène El Niño augmentant fortement la surface des eaux en 2012. Dès lors, la tendance est en pleine descente.

Tous ces résultats nous font comprendre la tendance générale de chaque lac. Il est maintenant intéressant d'analyser plus en détail ces dynamiques.

5.2 Evolution saisonnière de la surface des eaux (annexe : SCRIPT 2, p37)

Dans la même optique que précédemment, il existe une version de donnée **JRC** pour l'historique mensuel. Mais après une série de tests, il s'avérait que trop de données manquaient afin de produire des résultats un minimum valides.

Il est aussi intéressant d'explorer les potentiels de GEE en prenant des collections d'images « brutes » sur lesquelles on peut appliquer nos propres indices. C'est pourquoi j'ai choisi d'utiliser le deuxième jeu de données présenté dans la partie 4.1.

J'ai commencé par effectuer un test avec la collection d'images issue du Landsat 7 « **USGS Landsat 7 Collection 1 Tier 1 TOA Reflectance** » vu que les premières images disponibles datent de 1999, mais les résultats donnaient excessivement d'images saccadées pour en tirer des conclusions valides par saison.

J'ai donc choisi le jeu de données de Landsat 8, ce qui réduit le nombre d'années analysables mais nous permet d'obtenir des résultats à priori valides.

5.2.1 BARINGO

Commençons avec le lac Baringo.

Contrairement aux deux autres lacs (nous le verrons pourquoi par la suite), nous pouvons calculer l'aire de surface hydrologique pour chacune des images de la collection.

En effet, après l'avoir filtrée en fonction de la région d'intérêt et de la couverture nuageuse, le nombre d'images par année reste assez élevé (96 images pour les 9 années de 2013 à 2021). De cette manière, cela nous permet aussi d'analyser facilement par la suite les tendances saisonnières avec Excel.

Il a fallu redéfinir notre **ROI** en fonction de l'image avec la plus grande étendue d'eau (datant du 1er mars 2021).

J'ai ensuite dessiné un polygone avec un léger écart, dont les coordonnées définissent notre nouvelle **ROI**.

Les images Landsat sont organisées en rang et colonnes (**WRS_PATH** & **WRS_ROW**) que l'on retrouve grâce aux métadonnées.

En fonction de l'identité de cette propriété pour notre **ROI** et d'un plafond de 20% pour la couverture nuageuse, j'ai pu filtrer la collection. Ensuite, j'ai défini une fonction permettant d'extraire chaque pixel représentant une présence d'eau grâce à l'indice **NDWI,** qui dans ce cas, contrairement au **mNDWI,** est insensible aux éventuels nuages aux alentours du lac mais compris dans la zone d'intérêt.

Afin de calculer l'indice, il faut prendre en compte les bandes vertes (**B3**) et proche-infrarouge (**B5**), étant donné les paramètres de l'équation du NDWI :

« NDWI = (VERT – PIR)/(VERT + PIR) » (Mcfeeters 1996).

Une fois un masque de la zone hydrologique créé, je peux appliquer la fonction permettant de calculer l'aire du masque créé précédemment.

Finalement, nous obtenons à nouveau une liste à laquelle on vient ajouter les dates respectives sur chaque élément afin de pouvoir les identifier simplement et de faciliter la suite sur Excel pour générer les graphiques.

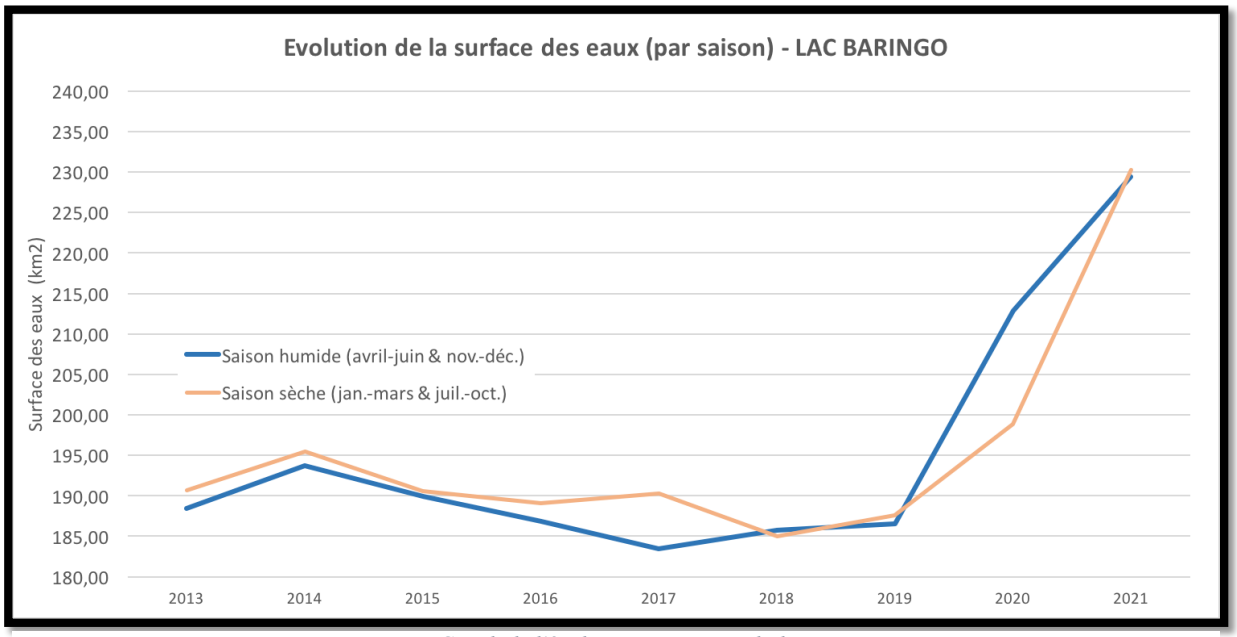

Figure 7: Graph de l'évolution par saison du lac Baringo

On comprend ici que dès 2019 le niveau des eaux a significativement augmenté. Étonnamment, dans la première partie, les surfaces des saisons sèches sont chaque année légèrement plus élevées que celles des saisons humides.

Cela est sans doute dû au biais théorique des périodes sèches et humides que nous avons déterminé au début. En effet, les chiffres sont issus des moyennes des mois appartenant à une saison ou une autre. Cependant, en pleine période de montée des eaux, notre saison humide montre un écart logique vis-à-vis de la saison sèche.

On peut tout de même déjà noter un taux de surface gagnée de +23,32% entre 2019 et 2021, ce qui est plutôt considérable.

Grâce à notre méthode de collecte de donnée spécifique à ce lac (avec les données Landsat), nous pouvons générer un autre graphique montrant l'étendue des différences de surfaces annuelles :

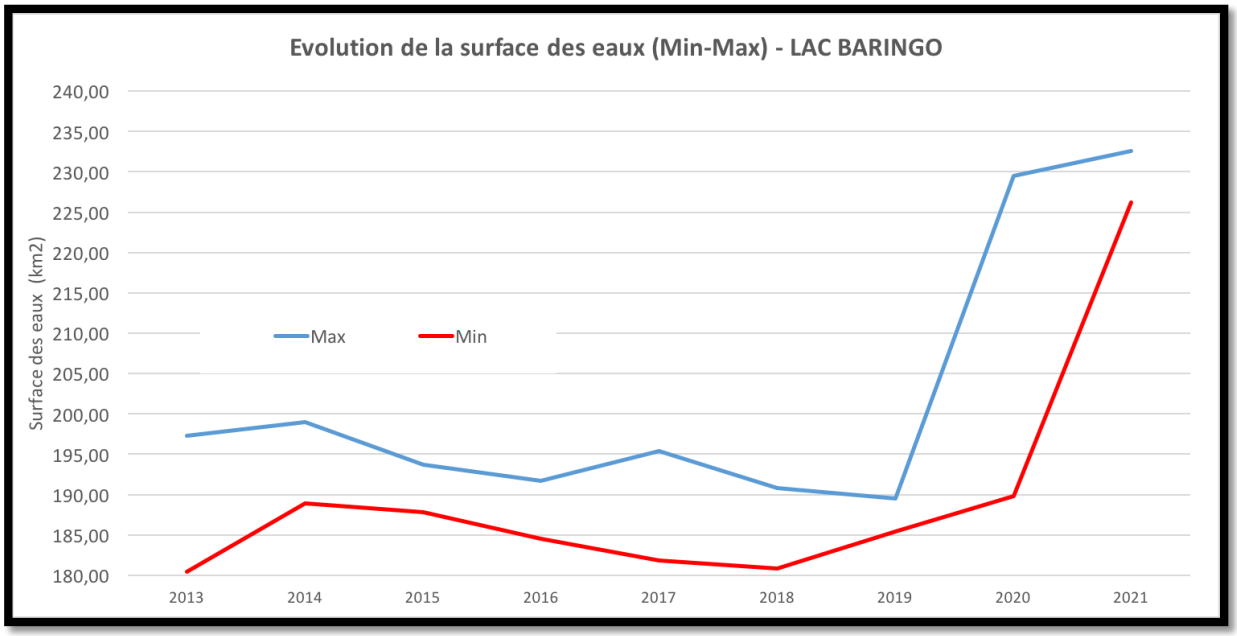

Figure 8: Graph de l'évolution Min/Max du lac Baringo

Il est intéressant de confirmer le fort dérangement dans la dynamique de l'évolution des eaux.

On voit ici que lors de la forte augmentation du niveau en 2020, l'étendue maximum de la surface était de 229,5 km2 (le 27 décembre) contre 189,78 km2 (le 10 janvier), soit un écart de 39,72 km2 (+20,9%) entre le début et la fin de l'année.

Pour se rendre compte spatialement de ces écarts, voici une représentation cartographique illustrant l'écart entre la plus faible et la plus forte étendue des eaux enregistrées entre 2013 et 2021 :

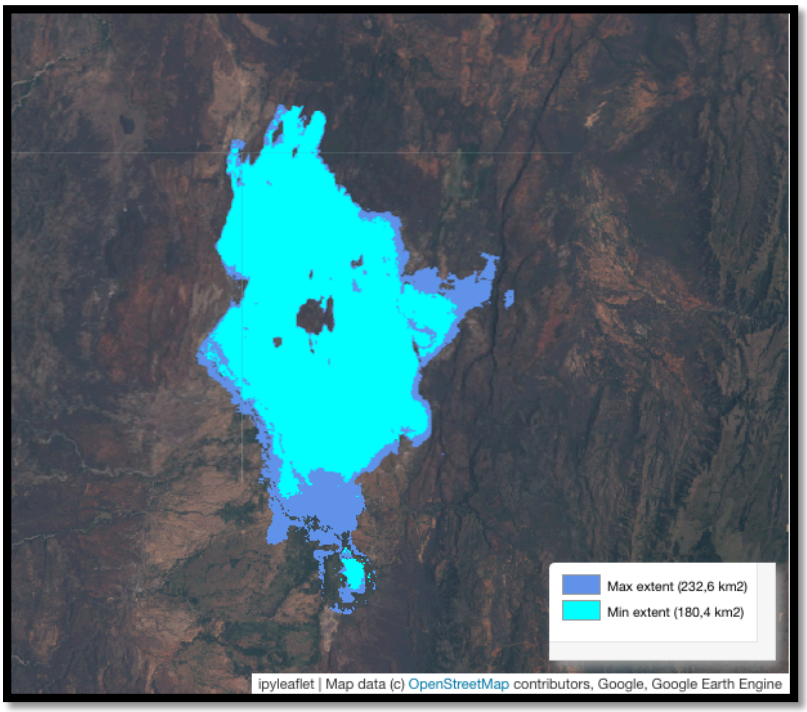

Figure 9: Carte Min/Max du lac Baringo (NDWI, Landsat)

5.2.2 TONLE SAP (annexe : SCRIPT 4, p45)

Pour évaluer les saisonnalités du lac Tonlé Sap, nous pourrions suivre théoriquement la même méthodologie.

En l'appliquant j'ai remarqué qu'elle n'était pas compatible car certaines données cruciales manquaient, et ce pour deux raisons :

- La zone d'intérêt étant beaucoup plus grande, les calculs demandent à couvrir une aire dont une seule image Landsat (sur qu'un seul **ROW** et qu'un seul **PATH**) ne suffit pas. Il nous faut donc assembler une multitude de clichés afin de couvrir la zone entière. Ce qu'il manquait par exemple pour l'année 2013 : aucune image ne couvrait la zone passant par le **WRS_ROW 51** et **WRS_PATH 127**.
- Le plafond de pourcentage de couverture du sol avait atteint sa limite, car en le baissant, le nombre d'images disponible n'était pas assez élevé pour combler les trous temporels, et en le laissant au même niveau, la densité des nuages empêchait la détection des eaux.

C'est pourquoi j'ai décidé d'utiliser le troisième jeu de données issus du capteur MSI du satellite Sentinel 2. Grâce à sa technologie, il est possible de créer des masques détectant les nuages et de les supprimer afin d'éviter tout dérangement de ces derniers[6].

Mais une fois cet algorithme appliqué, un dernier problème est à régler. Selon les images générées, quelques parties ne semblent pas être considérées par l'indice **NDWI**. J'ai donc décidé d'utiliser un autre indice développé après le premier : le **mNDWI** (pour modified Normalized Difference Water Index).

Cette deuxième génération se sert de la bande infrarouge à courtes ondes (**SWIR**) qui s'avère capable de refléter d'autres subtiles caractéristiques de l'eau, notamment moins sensible aux concentrations de sédiments (Huang et al. 2018) ce qui semble d'après les images être la source du problème. Les résultats sont convaincants.

La démarche reste à la base la même que celle utilisée pour le lac Baringo, il a cependant d'abord fallu rajouter les fonctions créant les différents masques de nuages (nuages, cirrus, ombres) pour ensuite créer une image médiane (**.median()**) des ensembles d'images appartenant aux mêmes groupes afin que les masques « anti-nuages » fassent effet et comblent les trous occasionnés.

Cette fonction **.median()** est idéale car elle va venir calculer les valeurs médianes de chaque bande de chaque image pour chaque pixel ; ce qui permet donc de réduire en une image saisonnière représentative des collections d'images filtrées en fonction des mois prédéfinis. Donc une fois le biais des nuages anéanti, j'ai divisé en deux groupes les mois les plus polarisés en termes de moyennes saisonnières théoriques (saison humide effective : de août à novembre ; saison sèche effective : de février à mai).

Il a fallu ensuite filtrer la collection pour ne prendre en compte que les images depuis 2019, car sous cette **ROI** incluant la zone forestière inondée, seulement 10 images étaient disponibles pour 2018 et surtout seulement entre le 17 et le 24 décembre, soit incomplètes pour effectuer un suivi saisonnier.

En comparaison au lac Baringo, ces temporalités sont nettement plus réduites mais nous verrons dans la prochaine partie que nous pouvons en tirer profit pour les années à venir.

[6] https://github.com/google/earthengine-community/blob/master/tutorials/sentinel-2-s2cloudless/index.ipynb, consulté le 15 décembre 2021.

Il faut finalement noter le changement des paramètres de la fonction de l'extraction de la couche hydrologique : c'est-à-dire substituer au calcul la bande B8 (**NIR**) pour la B11 (**SWIR1**) et ainsi utiliser le mNDWI à la place du NDWI.
La suite du script reste plus ou moins identique à quelques adaptations près.

Nous obtenons finalement une liste des aires de surfaces des eaux à laquelle on rajoute les labels des années respectives ainsi que des saisons.
L'opération est répétée pour la saison sèche. Voici les résultats :

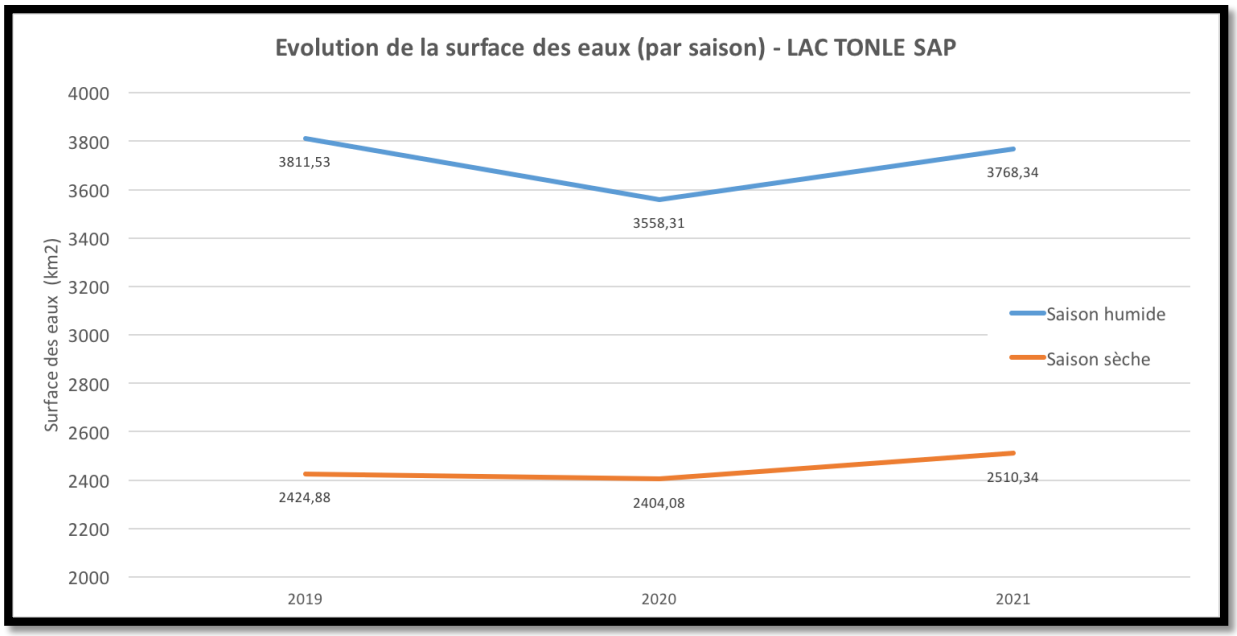

Figure 10: Graph de l'évolution par saison pour le lac Tonlé Sap (Sentinel)

On peut alors voir la différence significative entre les moyennes des saisons sèches comparées aux saisons humides. En moyenne sur ces trois ans, une hausse de +51,8% de la surface des eaux se produit annuellement.
De manière globale, une relative légère baisse s'est produite en 2020, mais la tendance semble s'être inversée pour 2021.

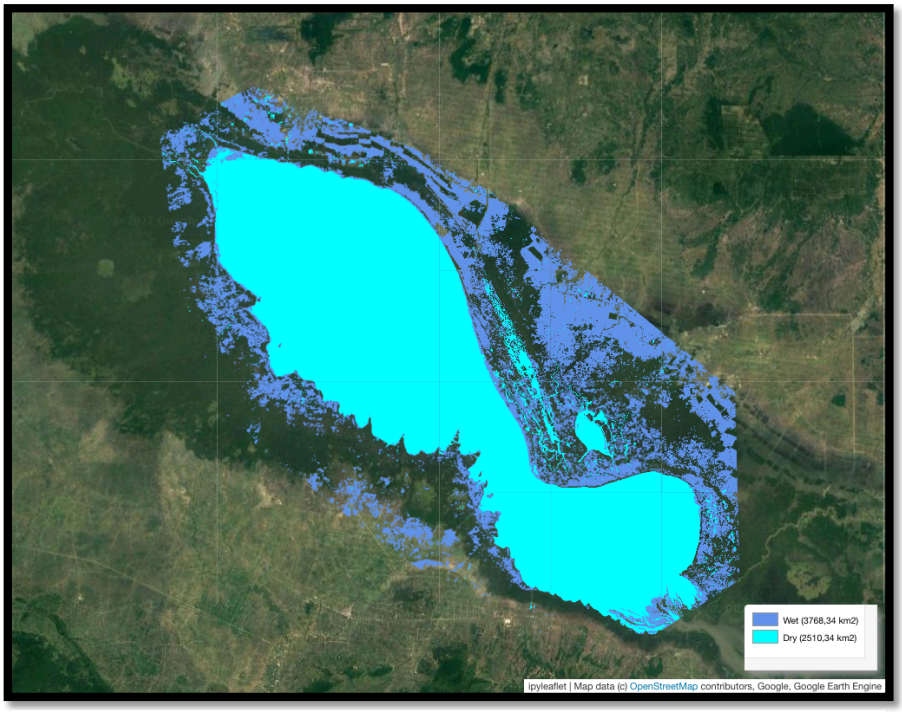

Figure 11: Carte saisonnière du lac Tonlé Sap, 2021 (mNDWI Sentinel)

5.2.3 SUESCA (annexe : SCRIPT 5, p53)

Pour la lagune de Suesca, nous allons procéder avec la même méthode que pour Tonlé Sap.

En effet, la couverture nuageuse trop élevée et l'incomplétude des données Landsat ne nous permettent pas de les utiliser.

A l'image de l'analyse précédente, celle-ci débute aussi en 2019, avec l'utilisation du même jeu de données Sentinel.

Il a donc fallu adapter le script en fonction de notre **ROI** pour Suesca.

Finalement, nos images disponibles par année sont plus nombreuses que celle issues de la collection de Landsat 8, mais nous n'en disposons que d'une partie réduite ne couvrant pas la totalité des mois pour chaque saison (beaucoup d'images pour certains mois de l'année).

Ce qui nous réduit l'analyse aux mois de janvier et février pour la saison sèche et aux mois d'octobre à décembre pour la saison humide. Ce trop peu d'images peut créer des trous dans les représentations cartographiques par le fait que le masque de nuage peut se voir non rempli par manque de donnée à certains endroits :

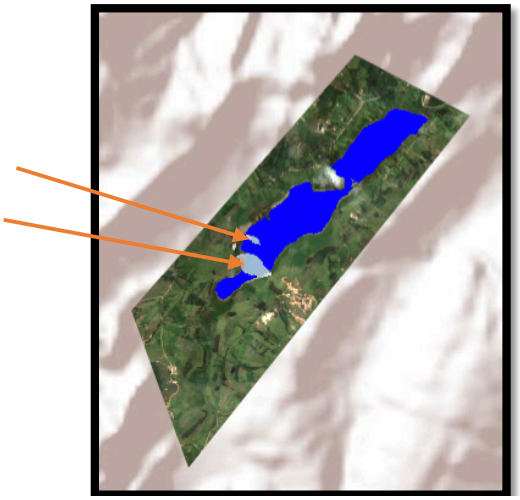

Figure 12: Capture d'écran du biais du masque nuageux

Mais ce biais semble impacter seulement la saison humide de l'année 2019 et une petite portion pour l'année 2020, et nous verrons dans la partie suivante qu'il sera contourné en effectuant une analyse annuelle.

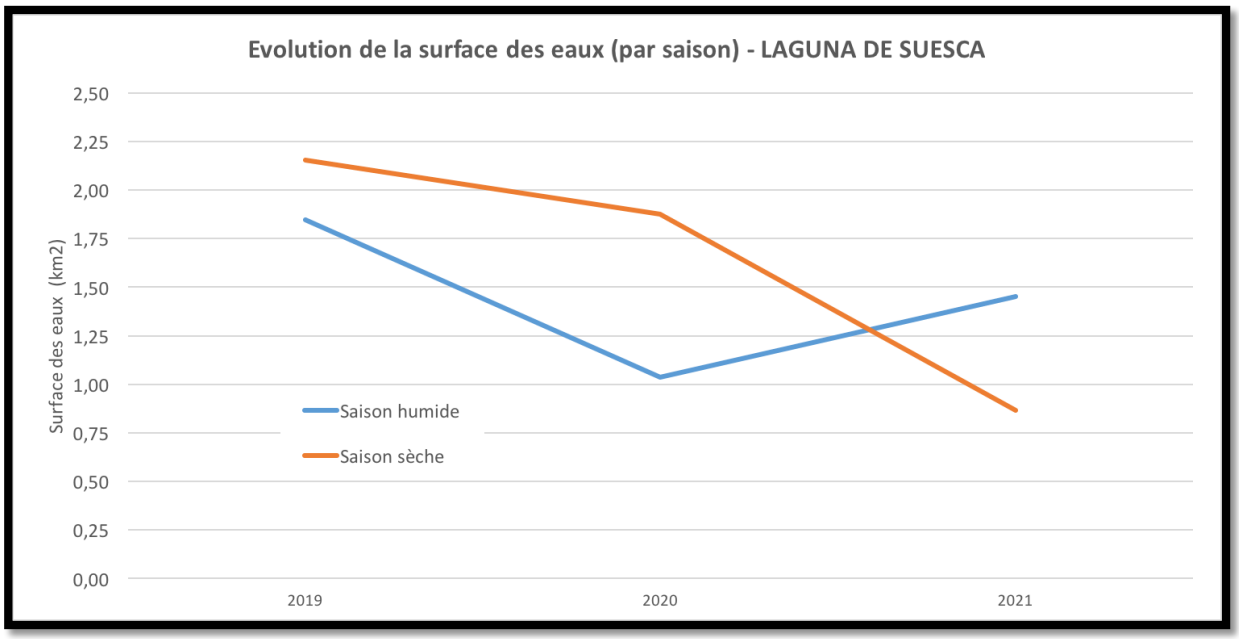

Figure 13: Graph de l'évolution saisonnière du lac de Suesca (Sentinel)

On observe ici une plus grande aire de surface pour la saison sèche pour les deux premières années. Ceci est surement dû aux biais expliqués plus haut : le manque de données pour prendre en compte la totalité des mois représentant chaque saison.

Malgré cela, on peut conclure un changement de régime de la dynamique saisonnière entre 2020 et 2021.

Il est intéressant aussi de noter que la tendance générale en est effectivement à une baisse générale de la surface des eaux entre 2019 et 2021.

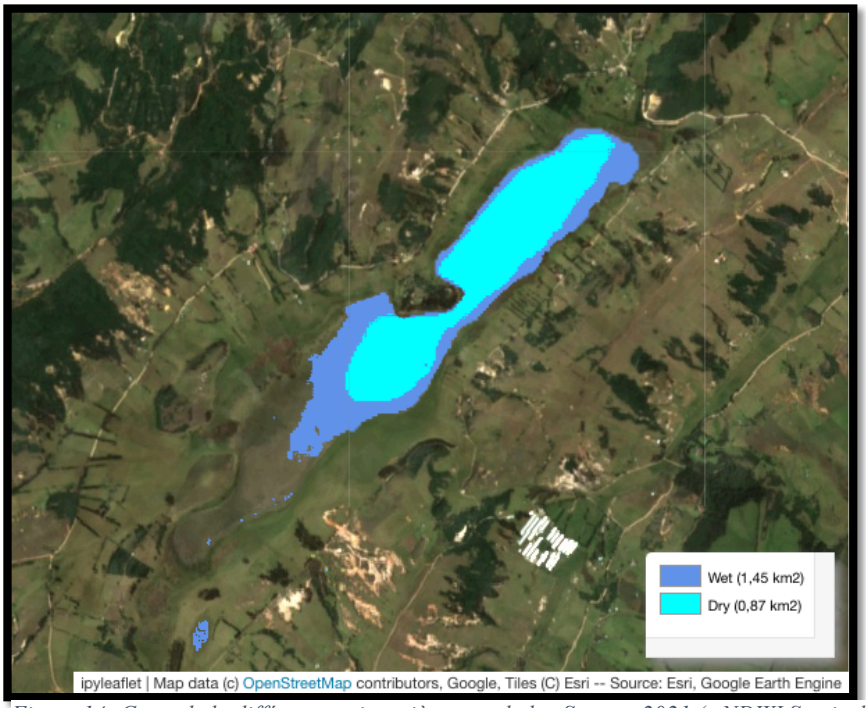

Figure 14: Carte de la différence saisonnière pour le lac Suesca, 2021 (mNDWI Sentinel)

Ces biais peuvent cependant être contrés en réalisant et analysant les médianes annuelles, ce qui permet d'agréger plus d'image et de « combler les trous ».
C'est ce que nous allons tenter de faire dans la prochaine partie.

5.3 Modèle d'analyse de suivi annuel applicable pour les années futures

Pour cette partie, nous allons donc récupérer les scripts générés pour les deux analyses saisonnières pour Tonlé Sap et Suesca, et les mettre à jour de sorte que l'on puisse générer un suivi annuel pour chacun des trois lacs.
Finalement, nous obtiendrons les moyennes annuelles des aires de surface des eaux, issues de données complètes valides grâce à l'application du masque de nuage non biaisé par le manque d'images.
Les trois nouveaux scripts générés sont donc adaptés et ne filtrent plus les collections d'images par saison, mais par année.
Les cartes produites montrent toutes la différence de surface entre les années 2019 et 2021.
Voici les résultats :

BARINGO (annexe : SCRIPT 3, p40) :

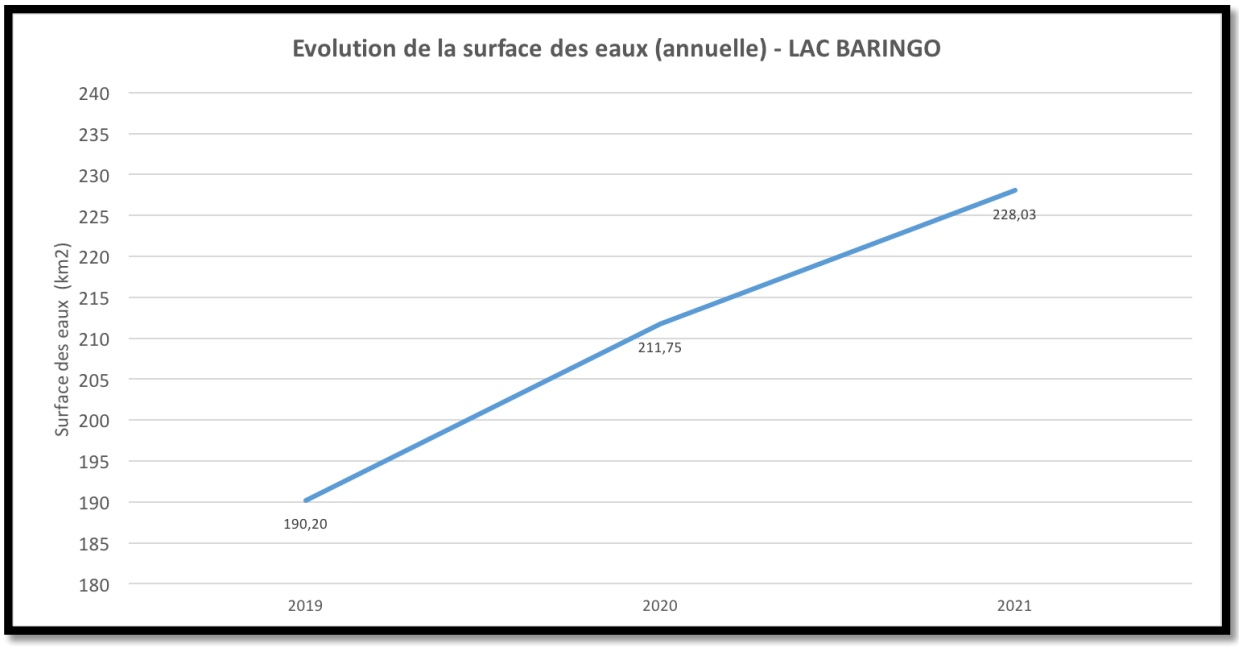

Figure 15: Graph de l'évolution annuelle du lac Baringo (Sentinel)

On témoigne de la nette augmentation de surface des eaux du lac Baringo de +11,33% entre 2019 et 2020 et de +7,69% entre 2020 et 2021. Soit tout de même +19,9% entre 2019 et 2021 ce qui représente un total de +37,84 km^2 de gain.

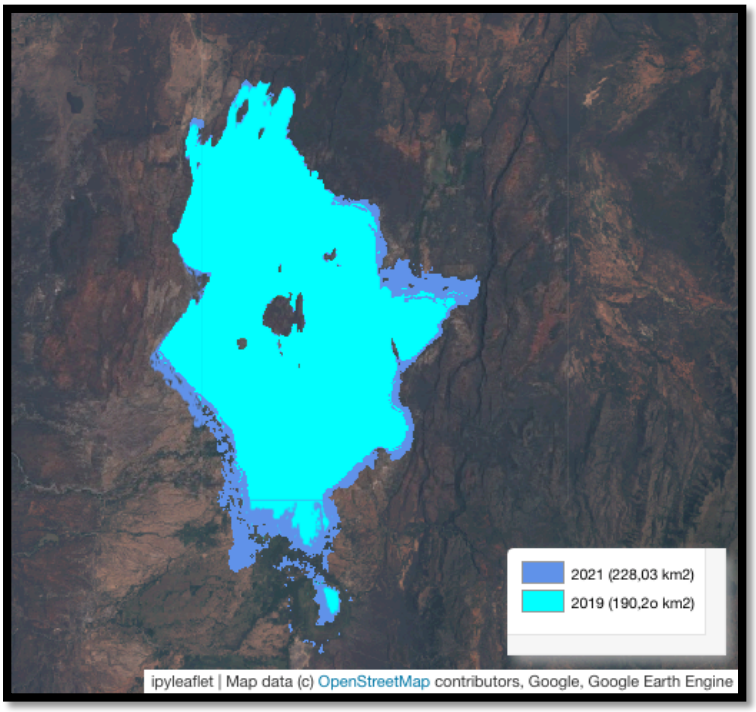

Figure 16: Carte de l'évolution 19/21 du lac Baringo (mNDWI Sentinel)

TONLE SAP (annexe : SCRIPT 4, p45):

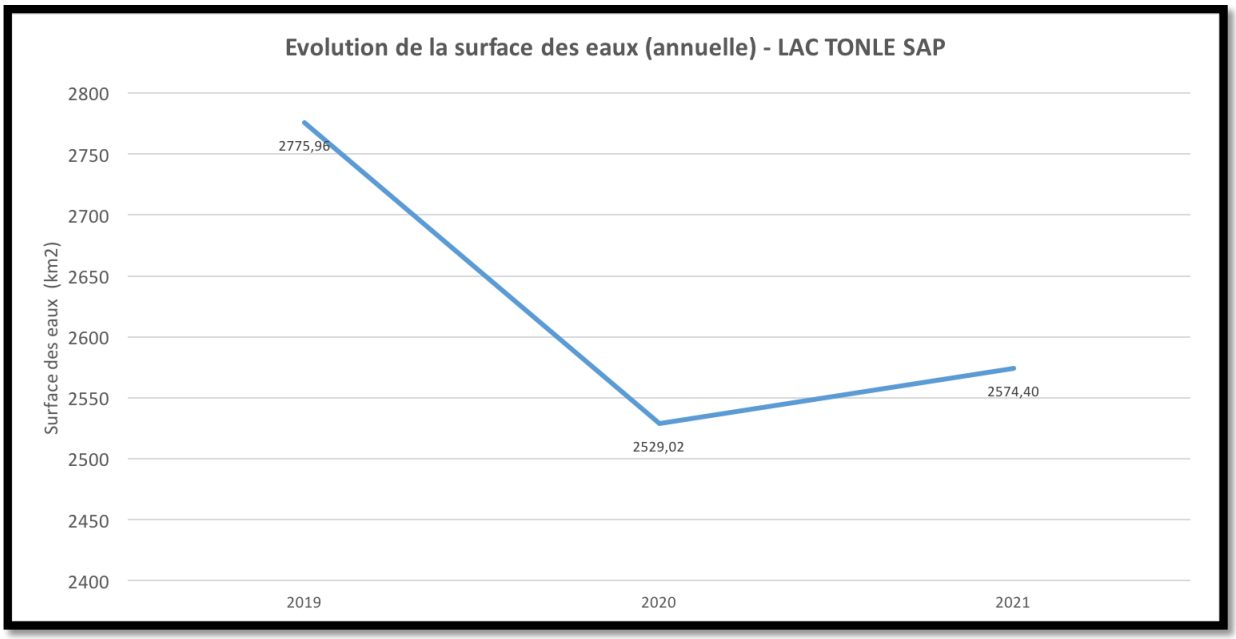

Figure 17: Graph de l'évolution annuelle du lac Tonlé Sap (Sentinel)

Une nette tendance à diminuer au cours de ces trois années pour le lac Tonlé Sap, même si entre 2020 et 2021, une augmentation de +1,79% s'est produite.

Entre ces trois années, les moyennes annuelles des surfaces ont chuté de -7,26%, soit -201,56 km^2 pour notre région d'intérêt sélectionnée.

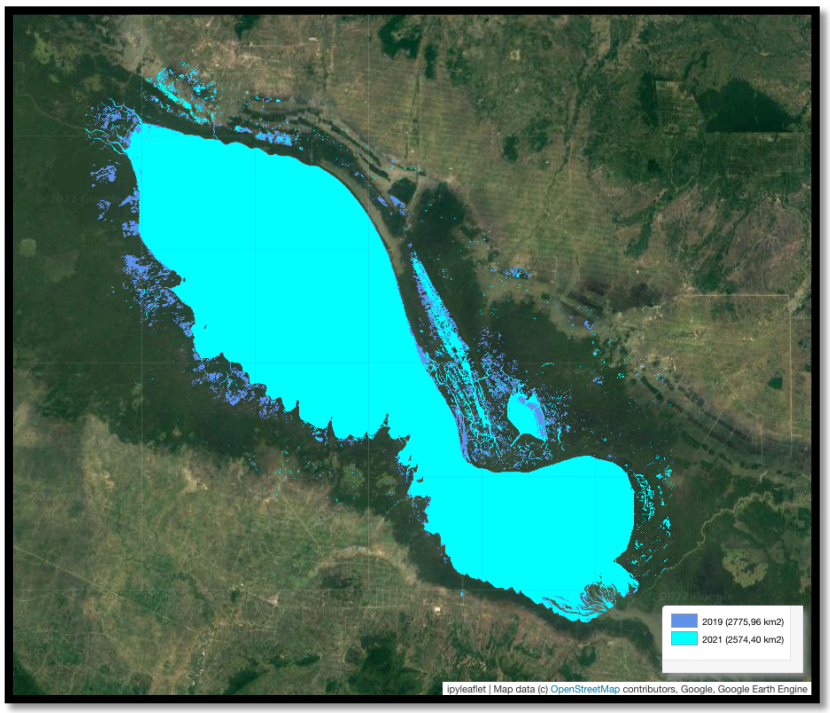

Figure 18: Carte de l'évolution 19/21 du lac Tonlé Sap (mNDWI Sentinel)

LAGUNA DE SUESCA (annexe : SCRIPT 5, p51) :

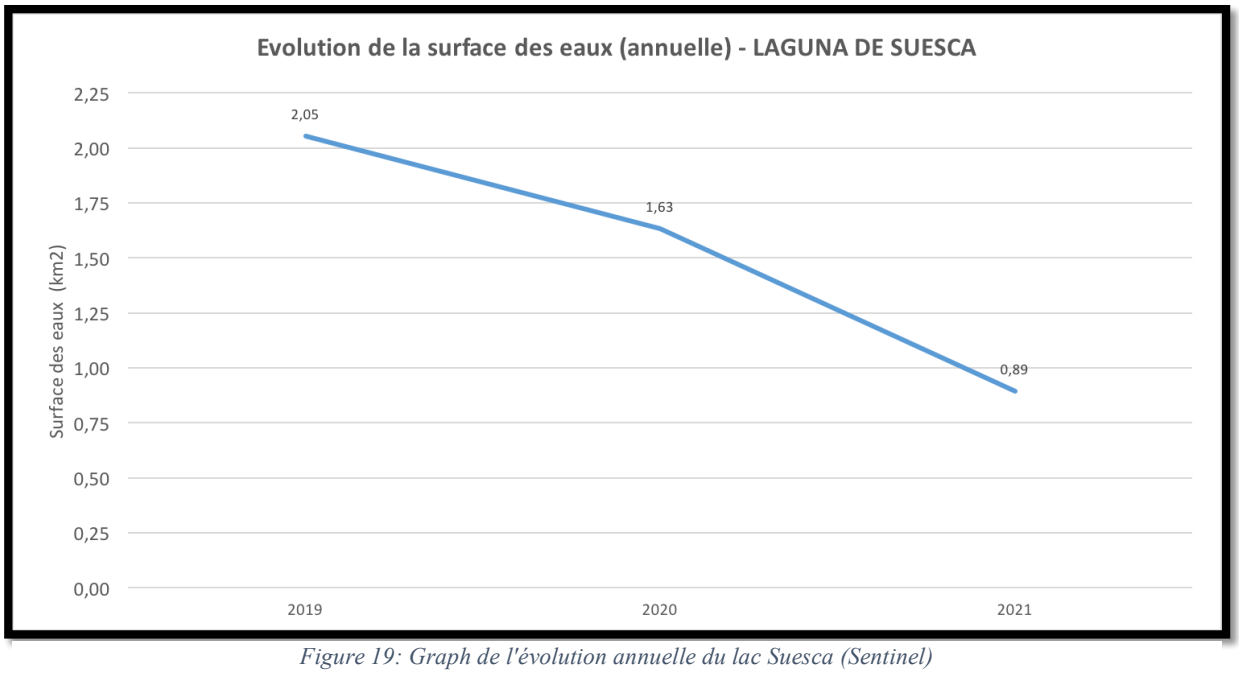

Figure 19: Graph de l'évolution annuelle du lac Suesca (Sentinel)

Pour la lagune de Suesca, on observe ainsi une tendance significative à une perte de surface. Entre 2019 et 2020, on enregistre -20,46% de pertes.

Et entre 2020 et 2021, -45,31%. En trois ans, la lagune a perdu -1,16 km^2, soit -56,5% de sa surface.

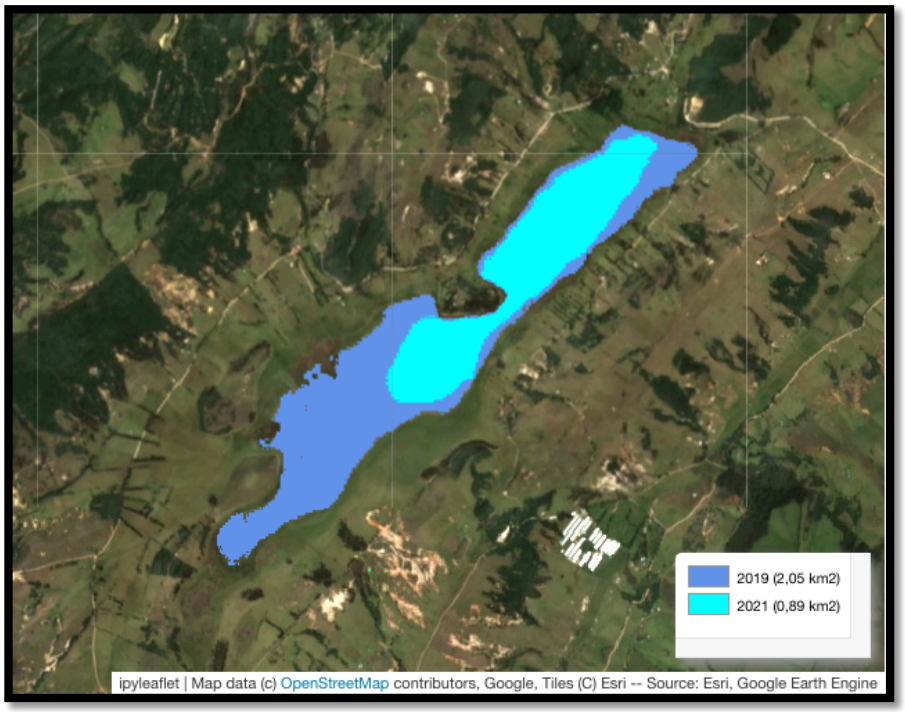

Figure 20: Carte de l'évolution 19/21 du lac Suesca (mNDWI Sentinel)

Grâce au masque de nuage et à l'agrégation des différentes images de chaque collection issus du jeu de donnée « **Sentinel-2 MSI: MultiSpectral Instrument, Level-2A** », cela nous offre des résultats robustes fiables quant au suivi de l'évolution annuelle.
Les scripts utilisés pour générer ces résultats sont relativement facilement ajustables. Ce processus d'analyse peut donc tout-à-fait être réutilisé dans le futur et les données sur le suivi ne feront que de s'agrandir au fur et à mesure que les années passent et que le satellite scanne la Terre.
Ces trois lacs disposent donc d'un système prêt à évaluer ses futures dynamiques de surfaces des eaux, système avec lequel il est tout-à-fait possible d'adapter à n'importe quel bassin hydrologique du monde.

6. DEVELOPPEMENT D'UNE APPLICATION WEB (annexe : SCRIPT 6, p60)

Afin de visualiser ces résultats et de pousser encore plus loin l'apprentissage et les possibilités de GEE, il est intéressant de développer une application web interactive. Pour ce faire, je vais me baser sur la méthode que j'avais utilisée pour le cours SDI dans le cadre du certificat de géomatique.
Je peux ainsi récupérer le script préalablement créé et l'adapter aux données de cette étude.

Afin de garder les interactions de l'application faciles, je choisis de mettre le moins d'options possibles, tout en essayant de la rendre intéressante.
C'est donc grâce à l'interface Heroku qui me permet d'héberger l'URL, que je peux faire le lien entre mon script et Internet. Voici l'adresse URL pour l'application, accessible librement :

https://geesimongenoud.herokuapp.com/

Basé sur le même jeu de donnée qu'au point précédent, je sélectionne chaque composition colorée pour chaque région, pour 2019 et pour 2021.
En faisant attention de prendre des **ROI** plus larges par soucis d'esthétique et en ajoutant l'outil **ts_inspector**, je peux créer en une seule application un outil de visualisation des différences de surfaces entre 2019 et 2021.

Il est donc possible de sélectionner les couches que l'on veut afficher et de se balader sur le globe librement ; et donc de choisir d'observer soit le lac Baringo, Tonlé Sap ou la lagune de Suesca.

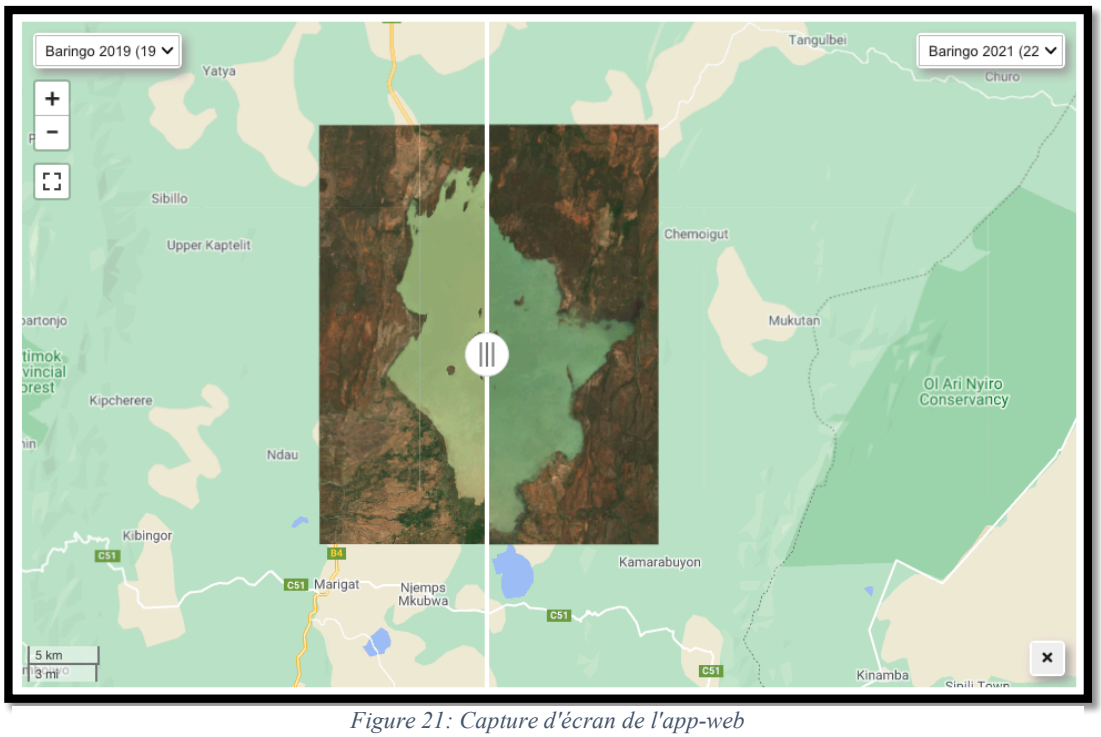

Figure 21: Capture d'écran de l'app-web

Évidemment, à l'image de la méthode précédente, les données peuvent être mises à jour. Donc en quelques lignes de code, nous pourrons ajouter les données futures et pousser l'étude dans le temps.

7. CONCLUSION

Pour conclure ce travail, nous avons vu que le développement d'un processus d'analyse de suivi des surfaces des eaux était possible grâce à Google Earth Engine.

En effet, c'est notamment la caractéristique de « cloud-computing » offrant la possibilité d'aller chercher des données dans de multiples bases issus de différents programmes satellitaires, qui fait la force de ce logiciel.

Nous avons donc pu développer plusieurs scripts capables de générer tous les résultats présentés au cours de cette étude, dont un modèle adaptable à un suivi futur des surfaces des bassins hydrologiques.

C'est finalement avec le développement d'une application web que nous pouvons aussi se rendre compte de l'évolution des surfaces des eaux.

Nous en concluons que l'état général est alarmant, et ce dans le contexte du dérèglement climatique.

Tous ces résultats sont cependant dépendants des choix subjectifs que j'ai effectué au cours de la mise en place des scripts, il va de soi qu'il faille prendre en compte une part de biais possible à différents niveaux de l'analyse.

J'espère tout de même que les produits de ce travail pourront servir peut-être de support à d'autres études, et qui sait, à améliorer notre connaissance du monde.

8. BIBLIOGRAPHIE

Amani M. *et al*. (2020). Google Earth Engine Cloud Computing Platform for Remote Sensing Big Data Applications: A Comprehensive Review, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 5326-5350.

Broich M., Kommareddy A., Stehman S. V. & Tulbure M. G. (2016). Surface water extent dynamics from three decades of seasonally continuous Landsat time series at subcontinental scale in a semi-arid region. *Remote Sensing of Environment*, Volume 178, 142-157.

Castellanos U., Francy L. (2005). Plan de desarrollo turístico sostenible para el municipio de Suesca (Cundinamarca) - Sustainable tourism development plan for the municipality of Suesca (Cundinamarca), Colombia. *Turismo & Sociedad*, 211-226.

Dixon M., Gorelick N., Hancher M., Ilyushchenko S., Moore R. & Thau D. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, 202, 18-27.

Huang, C., Chen, Y., Zhang, S., & Wu, J. (2018). Detecting, Extracting, and Monitoring Surface Water From Space Using Optical Sensors: A Review. *Reviews of Geophysics*, 56(2), 333-360.

Kiage L., Liu K., Walker N., Lam N., & Huh O. (2007). Recent land-cover/use change associated with land degradation in the Lake Baringo catchment, Kenya, East Africa: Evidence from Landsat TM and ETM. *International Journal of Remote Sensing*, 28(19), 4285-4309.

McFeeters S. K. (1996). The use of the normalized difference water index (NDWI) in the delineation of open water features. *International Journal of Remote Sensing*, 17(7), 1425–1432.

Ochieng R., Recha C., Bebe B.O. & Ogendi G.M. (2017). Rainfall Variability and Droughts in the Drylands of Baringo County, Kenya. *Open Access Library Journal*, 4: e3827.

Pekel JF., Cottam A., Gorelick N. et al. (2016). High-resolution mapping of global surface water and its long-term changes. *Nature,* 540, 418–422.

Petrakis R. E., Soulard C. E. & Walker J. J. (2020). Implementation of a Surface Water Extent Model in Cambodia using Cloud-Based Remote Sensing. *Remote Sensing (Basel, Switzerland)*, 12(6), 984.

Rebelo L.M., Finlayson C.M., & Nagabhatla, N. (2009). Remote sensing and GIS for wetland inventory, mapping and change analysis. *Journal of Environmental Management*, 90, 2144-2153.

Wen X. N.G., Edward P. (2021). Shrinking Tonlé Sap and the recent intensification of sand mining in the Cambodian Mekong River. *Science of The Total Environment*, Volume 777, 2021.

Wu Q., (2020). geemap: A Python package for interactive mapping with Google Earth Engine. *The Journal of Open Source Software*, 5(51), 2305.

Wu Q., Lane C., Li X., Zhao K., Zhou Y., Clinton N. & Lang M. (2019). Integrating LiDAR data and multi-temporal aerial imagery to map wetland inundation dynamics using Google Earth Engine. *Remote Sensing of Environment*, 228, 1-13.

9. ANNEXES

```
#----------------------------------SCRIPT 1: JRC ANNUEL-----------------------------------
```

In [ ]:

```
#importer les librairies utiles

import os
import geemap
import ee
```

In [ ]:

```
#geemap.update_package()
```

In [ ]:

```
#définir les roi des 3 lacs

roi_ba = ee.Geometry.Polygon([[[36.001087, 0.705197],
   [35.984546, 0.600005],
   [36.027803, 0.435705],
   [36.126273, 0.438526],
   [36.200758, 0.616623],
   [36.179423, 0.739859],
   [36.037956, 0.749635],
   [36.001087, 0.705197]]])

roi_to = ee.Geometry.Polygon([[[103.620294, 13.276684],
   [103.77034, 13.354224],
   [104.096705, 13.227225],
   [104.530051, 12.879877],
   [104.53701, 12.590711],
   [104.314043, 12.450951],
   [104.000031, 12.626352],
   [103.673644, 12.876603],
   [103.620294, 13.276684]]])

roi_su = ee.Geometry.Polygon([[[-73.806998, 5.173556],
   [-73.798954, 5.151531],
   [-73.75932, 5.200934],
   [-73.77549, 5.209497],
   [-73.806998, 5.173556]]])

features = [
    ee.Feature(roi_ba,  {'name': 'Baringo Lake'}),
    ee.Feature(roi_to,  {'name': 'Tonle Sap'}),
    ee.Feature(roi_su,  {'name': 'Laguna de Suesca'})
]

roifc = ee.FeatureCollection(features)
```

In [ ]:

```
#créer une collection

collection = ee.ImageCollection('JRC/GSW1_3/YearlyHistory')
vis_gsw = {
    'bands': ['waterClass'],
    'min': 0,
    'max': 3,
    'palette': ['cccccc', 'ffffff', '99d9ea', '0000ff']
}
```

```
collection.size().getInfo()
```

In [ ]:

```
#créer image d'une collection baringo

filmBA = {
  'dimensions': 500,
  'region': roi_ba,
  'crs': 'EPSG:3857',
  'min': 0,
  'max': 3,
  'palette': ['cccccc', 'ffffff', '99d9ea', '0000ff'],
}

print(collection.getFilmstripThumbURL(filmBA))
```

In [ ]:

```
#créer image d'une collection tonle sap

filmTO = {
  'dimensions': 500,
  'region': roi_to,
  'crs': 'EPSG:3857',
  'min': 0,
  'max': 3,
  'palette': ['cccccc', 'ffffff', '99d9ea', '0000ff'],
}

print(collection.getFilmstripThumbURL(filmTO))
```

In [ ]:

```
#créer image d'une collection laguna de suesca

filmSU = {
  'dimensions': 500,
  'region': roi_su,
  'crs': 'EPSG:3857',
  'min': 0,
  'max': 3,
  'palette': ['cccccc', 'ffffff', '99d9ea', '0000ff'],
}

print(collection.getFilmstripThumbURL(filmSU))
```

In [ ]:

```
#Créer stats Baringo

collba = collection.filter(ee.Filter.gte('year', 2000)).map(lambda img:
img.selfMask().clip(roi_ba))

def cal_area_ba(img):
    waterClass = img.select('waterClass')
    areaImage = img.pixelArea().divide(1e6).addBands(waterClass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': roi_ba,
      'scale': 30,
      'maxPixels': 1e9
```

```
    })
    return img.set({'water_area': stats})

reduction_results_ba = collba.map(cal_area_ba)

statsba = ee.List(reduction_results_ba.aggregate_array('water_area'))
yearsba = ee.List(reduction_results_ba.aggregate_array('year'))
finalstatsba = statsba.zip(yearsba)
finalstatsba.getInfo()
```

In [ ]:

```
#Créer stats Tonle Sap

collto = collection.filter(ee.Filter.gte('year', 2000)).map(lambda img:
img.selfMask().clip(roi_to))

def cal_area_to(img):
    waterClass = img.select('waterClass')
    areaImage = img.pixelArea().divide(1e6).addBands(waterClass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': roi_to,
      'scale': 30,
      'maxPixels': 1e9
    })
    return img.set({'water_area': stats})

reduction_results_to = collto.map(cal_area_to)

statsto = ee.List(reduction_results_to.aggregate_array('water_area'))
yearsto = ee.List(reduction_results_to.aggregate_array('year'))
finalstatsto = statsto.zip(yearsto)
finalstatsto.getInfo()
```

In [ ]:

```
#Créer stats Laguna de Suesca

collsu = collection.filter(ee.Filter.gte('year', 2000)).map(lambda img:
img.selfMask().clip(roi_su))

def cal_area_su(img):
    waterClass = img.select('waterClass')
    areaImage = img.pixelArea().divide(1e6).addBands(waterClass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': roi_su,
      'scale': 30,
      'maxPixels': 1e9
    })
    return img.set({'water_area': stats})

reduction_results_su = collsu.map(cal_area_su)
```

```
statssu = ee.List(reduction_results_su.aggregate_array('water_area'))
yearssu = ee.List(reduction_results_su.aggregate_array('year'))
finalstatssu = statssu.zip(yearssu)
finalstatssu.getInfo()
```

```
#----------------------------------SCRIPT 2: BARINGO SAISONS-------------
---------------------
```

In [ ]:

```
#importer les librairies utiles

import os
import geemap
import ee
```

In [ ]:

```
#geemap.update_package()
```

In [ ]:

```
#définir nouvelle roi pour baringo

roi_ba = ee.Geometry.Polygon([[[36.014251, 0.687962],
   [36.01964, 0.716279],
   [36.042607, 0.737815],
   [36.078199, 0.740942],
   [36.099526, 0.729917],
   [36.094446, 0.704853],
   [36.108309, 0.686524],
   [36.134839, 0.684272],
   [36.138188, 0.663977],
   [36.154946, 0.651468],
   [36.180029, 0.644709],
   [36.179509, 0.62347],
   [36.147641, 0.582614],
   [36.147418, 0.549394],
   [36.135416, 0.532821],
   [36.118008, 0.526524],
   [36.112094, 0.501258],
   [36.116373, 0.470073],
   [36.101121, 0.450215],
   [36.073854, 0.455284],
   [36.062996, 0.48033],
   [36.03783, 0.485635],
   [36.033179, 0.507035],
   [36.027608, 0.52135],
   [36.028699, 0.545695],
   [35.998659, 0.590175],
   [36.011058, 0.625683],
   [36.007195, 0.653589],
   [36.014251, 0.687962]]])
```

In [ ]:

```
#Créer stats Baringo Landsat pour saisons (image par image)

#filtrer image collection
icba = ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA')\
    .filter(ee.Filter.eq('WRS_PATH', 169))\
    .filter(ee.Filter.eq('WRS_ROW', 60))\
    .filterMetadata('CLOUD_COVER', 'less_than', 20)

#clip
icba_new = icba.map(lambda img: img.clip(roi_ba))

#def fontion pour extraire ndwi
def extract_water(img):
    ndwi_threshold = 0
```

```
    ndwi_image = img.normalizedDifference(['B3', 'B5'])
    water_image = ndwi_image.gt(ndwi_threshold).selfMask()
    return water_image

#appliquer fonction extraction ndwi
ndwi = icba_new.map(extract_water)

#stats saison (aires)
def area_season(img):
    wclass = img.select('nd')
    areaImage = img.pixelArea().divide(1e6).addBands(wclass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': roi_ba,
      'scale': 30,
      'maxPixels': 1e9
    })
    return img.set({'season_area': stats})

res_season = ndwi.map(area_season)

list1 = ee.List(res_season.aggregate_array('season_area').getInfo())
list2 = ee.List(icba_new.aggregate_array('DATE_ACQUIRED').getInfo())

final_list = list1.zip(list2)
final_list.getInfo()
```

In [ ]:

```
#définir nouvelle roi pour créer map

roi_map = ee.Geometry.Polygon([[[35.828647, 0.370185],
   [35.828647, 0.827958],
   [36.360828, 0.827958],
   [36.360828, 0.370185],
   [35.828647, 0.370185]]])
```

In [ ]:

```
#ajouter layers à la map

ba_rgb = ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA')\
    .filter(ee.Filter.eq('WRS_PATH', 169))\
    .filter(ee.Filter.eq('WRS_ROW', 60))\
    .filterMetadata('CLOUD_COVER', 'less_than',
20).median().clip(roi_map).select(['B4', 'B3', 'B2'])

trueColor432Vis = {
  'min': 0.0,
  'max': 0.3,}

Map.addLayer(ba_rgb, trueColor432Vis, 'ba_rgb', True,)

ba_max = ndwi.filterMetadata('system:index', 'equals',
'LC08_169060_20211112').first()
Map.addLayer(ba_max, {'palette': '6495ED'}, 'ba_max', True)

ba_min = ndwi.first()
```

```
Map.addLayer(ba_min, {'palette': '00FFFF'}, 'ba_min', True)

legend_dict = {
    'Max extent (232,6 km2)': '6495ED',
    'Min extent (180,4 km2)': '00FFFF'}



Map.add_legend(legend_title="", legend_dict=legend_dict)
```

In [ ]:

```
#afficher la map

Map = geemap.Map()
Map.setCenter(36.07, 0.61, 11)
Map
```

```
#----------------------------------SCRIPT 3: BARINGO SENTINEL ANNUEL------
-----------------------------
```

In [ ]:

```
#importer les librairies utiles

import os
import geemap
import ee
```

In [ ]:

```
#initialiser la librairie

ee.Initialize()
```

In [ ]:

```
#geemap.update_package()
```

In [ ]:

```
#définir nouvelle roi et valeurs pour baringo

AOI = ee.Geometry.Polygon([[[36.014251, 0.687962],
   [36.01964, 0.716279],
   [36.042607, 0.737815],
   [36.078199, 0.740942],
   [36.099526, 0.729917],
   [36.094446, 0.704853],
   [36.108309, 0.686524],
   [36.134839, 0.684272],
   [36.138188, 0.663977],
   [36.154946, 0.651468],
   [36.180029, 0.644709],
   [36.179509, 0.62347],
   [36.147641, 0.582614],
   [36.147418, 0.549394],
   [36.135416, 0.532821],
   [36.118008, 0.526524],
   [36.112094, 0.501258],
   [36.116373, 0.470073],
   [36.101121, 0.450215],
   [36.073854, 0.455284],
   [36.062996, 0.48033],
   [36.03783, 0.485635],
   [36.033179, 0.507035],
   [36.027608, 0.52135],
   [36.028699, 0.545695],
   [35.998659, 0.590175],
   [36.011058, 0.625683],
   [36.007195, 0.653589],
   [36.014251, 0.687962]]])
START_DATE = '2019'
END_DATE = '2022'
CLOUD_FILTER = 60
CLD_PRB_THRESH = 40
NIR_DRK_THRESH = 0.15
CLD_PRJ_DIST = 2
BUFFER = 100
```

In [ ]:

```
#----------------------------------DEBUT DU MASQUE NUAGEUX----------------
------------------
```

```
def get_s2_sr_cld_col(aoi, start_date, end_date):
    # Import and filter S2 SR.
    s2_sr_col = (ee.ImageCollection('COPERNICUS/S2_SR')
        .filterBounds(aoi)
        .filterDate(start_date, end_date)
        .filter(ee.Filter.lte('CLOUDY_PIXEL_PERCENTAGE', CLOUD_FILTER)))

    # Import and filter s2cloudless.
    s2_cloudless_col =
(ee.ImageCollection('COPERNICUS/S2_CLOUD_PROBABILITY')
        .filterBounds(aoi)
        .filterDate(start_date, end_date))

    # Join the filtered s2cloudless collection to the SR collection by the
'system:index' property.
    return ee.ImageCollection(ee.Join.saveFirst('s2cloudless').apply(**{
        'primary': s2_sr_col,
        'secondary': s2_cloudless_col,
        'condition': ee.Filter.equals(**{
            'leftField': 'system:index',
            'rightField': 'system:index'
        })
    }))
```

In [ ]:

```
s2_sr_cld_col = get_s2_sr_cld_col(AOI, START_DATE, END_DATE)
```

In [ ]:

```
def add_cloud_bands(img):
    # Get s2cloudless image, subset the probability band.
    cld_prb = ee.Image(img.get('s2cloudless')).select('probability')

    # Condition s2cloudless by the probability threshold value.
    is_cloud = cld_prb.gt(CLD_PRB_THRESH).rename('clouds')

    # Add the cloud probability layer and cloud mask as image bands.
    return img.addBands(ee.Image([cld_prb, is_cloud]))
```

In [ ]:

```
def add_shadow_bands(img):
    # Identify water pixels from the SCL band.
    not_water = img.select('SCL').neq(6)

    # Identify dark NIR pixels that are not water (potential cloud shadow
pixels).
    SR_BAND_SCALE = 1e4
    dark_pixels =
img.select('B8').lt(NIR_DRK_THRESH*SR_BAND_SCALE).multiply(not_water).renam
e('dark_pixels')

    # Determine the direction to project cloud shadow from clouds (assumes
UTM projection).
    shadow_azimuth =
ee.Number(90).subtract(ee.Number(img.get('MEAN_SOLAR_AZIMUTH_ANGLE')));

    # Project shadows from clouds for the distance specified by the
CLD_PRJ_DIST input.
    cld_proj =
(img.select('clouds').directionalDistanceTransform(shadow_azimuth,
CLD_PRJ_DIST*10)
```

```
        .reproject(**{'crs': img.select(0).projection(), 'scale': 100})
        .select('distance')
        .mask()
        .rename('cloud_transform'))

    # Identify the intersection of dark pixels with cloud shadow
projection.
    shadows = cld_proj.multiply(dark_pixels).rename('shadows')

    # Add dark pixels, cloud projection, and identified shadows as image
bands.
    return img.addBands(ee.Image([dark_pixels, cld_proj, shadows]))
```

In [ ]:

```
def add_cld_shdw_mask(img):
    # Add cloud component bands.
    img_cloud = add_cloud_bands(img)

    # Add cloud shadow component bands.
    img_cloud_shadow = add_shadow_bands(img_cloud)

    # Combine cloud and shadow mask, set cloud and shadow as value 1, else
0.
    is_cld_shdw =
img_cloud_shadow.select('clouds').add(img_cloud_shadow.select('shadows')).g
t(0)

    # Remove small cloud-shadow patches and dilate remaining pixels by
BUFFER input.
    # 20 m scale is for speed, and assumes clouds don't require 10 m
precision.
    is_cld_shdw = (is_cld_shdw.focalMin(2).focalMax(BUFFER*2/20)
        .reproject(**{'crs': img.select([0]).projection(), 'scale': 20})
        .rename('cloudmask'))

    # Add the final cloud-shadow mask to the image.
    return img_cloud_shadow.addBands(is_cld_shdw)
```

In [ ]:

```
def apply_cld_shdw_mask(img):
    # Subset the cloudmask band and invert it so clouds/shadow are 0, else
1.
    not_cld_shdw = img.select('cloudmask').Not()

    # Subset reflectance bands and update their masks, return the result.
    return img.select('B.*').updateMask(not_cld_shdw)

#----------------------------------FIN DU MASQUE NUAGEUX-----------------
-----------------
```

In [ ]:

```
s2_sr = s2_sr_cld_col.map(add_cld_shdw_mask).map(apply_cld_shdw_mask)
```

In [ ]:

```
#changer date system pour mieux filtrer après, + clip

baringocoll_new = s2_sr.map(lambda img: img.set({"DATE":
ee.Date(img.get("system:time_start"))\
                                        .format("YYYY-dd-MM")})).clip(AOI))
```

In [ ]:

```
#Stats annuelles
```

```
def yearly(year):
    start_date = ee.Date.fromYMD(year, 1, 1)
    end_date = start_date.advance(1, 'year')

    image = baringocoll_new.filterDate(start_date, end_date).median()
    return image

#definir liste années
start_year = 2019
end_year = 2021
years = ee.List.sequence(start_year, end_year)
year_list = years.getInfo()

#assembler années & collection saison
images = years.map(yearly)

#en faire une image collection
yearlycoll = ee.ImageCollection(images)

#def fontion pour extraire ndwi
def extract_water(img):
    mndwi_threshold = 0
    mndwi_image = img.normalizedDifference(['B3', 'B11'])
    water_image = mndwi_image.gt(mndwi_threshold).selfMask()
    return water_image

#appliquer fonction extraction ndwi
mndwi_yearly = yearlycoll.map(extract_water)

#stats saison
def area_season(img):
    wclass = img.select('nd')
    areaImage = img.pixelArea().divide(1e6).addBands(wclass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': AOI,
      'scale': 10,
      'maxPixels': 1e9
    })
    return img.set({'season_area': stats})

res_yearly = mndwi_yearly.map(area_season)

res_list = ee.List(res_yearly.aggregate_array('season_area'))

final = res_list.zip(year_list)

final.getInfo()
```

In [ ]:

```
#definir nouvelle roi pour la map

roi_map = ee.Geometry.Polygon([[[35.828647, 0.370185],
   [35.828647, 0.827958],
```

```
   [36.360828, 0.827958],
   [36.360828, 0.370185],
   [35.828647, 0.370185]]])
```

In [ ]:

```
#ajouter layer a la map

ba_rgb = ee.ImageCollection('LANDSAT/LC08/C01/T1_TOA')\
    .filter(ee.Filter.eq('WRS_PATH', 169))\
    .filter(ee.Filter.eq('WRS_ROW', 60))\
    .filterMetadata('CLOUD_COVER', 'less_than',
20).median().clip(roi_map).select(['B4', 'B3', 'B2'])

trueColor432Vis = {
  'min': 0.0,
  'max': 0.3,}

Map.addLayer(ba_rgb, trueColor432Vis, 'ba_rgb', True,)

mndwi_yearly_img21 = mndwi_yearly.filterMetadata('system:index', 'equals',
'2')
Map.addLayer(mndwi_yearly_img21, {'palette': '6495ED'},
'mndwi_yearly_img21', False)

mndwi_yearly_img19 = mndwi_yearly.filterMetadata('system:index', 'equals',
'0')
Map.addLayer(mndwi_yearly_img19, {'palette': '00FFFF'},
'mndwi_yearly_img19', False)


legend_dict = {
    '2021 (228,03 km2)': '6495ED',
    '2019 (190,2o km2)': '00FFFF'}

Map.add_legend(legend_title="", legend_dict=legend_dict)
```

In [ ]:

```
#afficher la map

Map = geemap.Map()
Map.setCenter(36.067, 0.65, 11)
Map
```

```
#----------------------------------SCRIPT 4: TONLE SAP SENTINEL-----------
------------------------
```

```
import os
import geemap
import ee
```

In [ ]:

```
#initialiser la librairie

ee.Initialize()
```

In [ ]:

```
#geemap.update_package()
```

In [ ]:

```
#definir nouvelle roi pour la map

AOIlarge = ee.Geometry.Polygon([[[103.620294, 13.276684],
   [103.77034, 13.354224],
   [104.096705, 13.227225],
   [104.530051, 12.879877],
   [104.53701, 12.590711],
   [104.314043, 12.450951],
   [104.000031, 12.626352],
   [103.673644, 12.876603],
   [103.620294, 13.276684]]])
```

In [ ]:

```
#définir nouvelle roi et valeurs pour tonle sap

AOI = ee.Geometry.Polygon([[[103.594031, 13.247261],
   [103.772345, 13.371756],
   [103.880567, 13.351228],
   [104.13652, 13.2391],
   [104.224176, 12.959331],
   [104.496784, 12.823686],
   [104.61171, 12.609349],
   [104.385482, 12.415391],
   [104.175083, 12.491261],
   [104.044597, 12.648881],
   [103.742939, 12.836166],
   [103.594031, 13.247261]]])
START_DATE = '2019'
END_DATE = '2022'
CLOUD_FILTER = 60
CLD_PRB_THRESH = 40
NIR_DRK_THRESH = 0.15
CLD_PRJ_DIST = 2
BUFFER = 100
```

In [ ]:

```
dataset =
ee.Image('JRC/GSW1_3/GlobalSurfaceWater').clip(AOI).select('seasonality').g
te(12).selfMask()
roitonle = dataset.reduceToVectors(**{'maxPixels' : 1e9})
```

In [ ]:

```
#----------------------------------DEBUT DU MASQUE NUAGEUX----------------
-------------------

def get_s2_sr_cld_col(aoi, start_date, end_date):
    # Import and filter S2 SR.
```

```
    s2_sr_col = (ee.ImageCollection('COPERNICUS/S2_SR')
        .filterBounds(aoi)
        .filterDate(start_date, end_date)
        .filter(ee.Filter.lte('CLOUDY_PIXEL_PERCENTAGE', CLOUD_FILTER)))

    # Import and filter s2cloudless.
    s2_cloudless_col =
(ee.ImageCollection('COPERNICUS/S2_CLOUD_PROBABILITY')
        .filterBounds(aoi)
        .filterDate(start_date, end_date))

    # Join the filtered s2cloudless collection to the SR collection by the
'system:index' property.
    return ee.ImageCollection(ee.Join.saveFirst('s2cloudless').apply(**{
        'primary': s2_sr_col,
        'secondary': s2_cloudless_col,
        'condition': ee.Filter.equals(**{
            'leftField': 'system:index',
            'rightField': 'system:index'
        })
    }))
```

In [ ]:

```
s2_sr_cld_col = get_s2_sr_cld_col(AOIlarge, START_DATE, END_DATE)
```

In [ ]:

```
def add_cloud_bands(img):
    # Get s2cloudless image, subset the probability band.
    cld_prb = ee.Image(img.get('s2cloudless')).select('probability')

    # Condition s2cloudless by the probability threshold value.
    is_cloud = cld_prb.gt(CLD_PRB_THRESH).rename('clouds')

    # Add the cloud probability layer and cloud mask as image bands.
    return img.addBands(ee.Image([cld_prb, is_cloud]))
```

In [ ]:

```
def add_shadow_bands(img):
    # Identify water pixels from the SCL band.
    not_water = img.select('SCL').neq(6)

    # Identify dark NIR pixels that are not water (potential cloud shadow
pixels).
    SR_BAND_SCALE = 1e4
    dark_pixels =
img.select('B8').lt(NIR_DRK_THRESH*SR_BAND_SCALE).multiply(not_water).renam
e('dark_pixels')

    # Determine the direction to project cloud shadow from clouds (assumes
UTM projection).
    shadow_azimuth =
ee.Number(90).subtract(ee.Number(img.get('MEAN_SOLAR_AZIMUTH_ANGLE')));

    # Project shadows from clouds for the distance specified by the
CLD_PRJ_DIST input.
    cld_proj =
(img.select('clouds').directionalDistanceTransform(shadow_azimuth,
CLD_PRJ_DIST*10)
        .reproject(**{'crs': img.select(0).projection(), 'scale': 100})
        .select('distance')
```

```
        .mask()
        .rename('cloud_transform'))

    # Identify the intersection of dark pixels with cloud shadow
projection.
    shadows = cld_proj.multiply(dark_pixels).rename('shadows')

    # Add dark pixels, cloud projection, and identified shadows as image
bands.
    return img.addBands(ee.Image([dark_pixels, cld_proj, shadows]))
```

In [ ]:

```
def add_cld_shdw_mask(img):
    # Add cloud component bands.
    img_cloud = add_cloud_bands(img)

    # Add cloud shadow component bands.
    img_cloud_shadow = add_shadow_bands(img_cloud)

    # Combine cloud and shadow mask, set cloud and shadow as value 1, else
0.
    is_cld_shdw =
img_cloud_shadow.select('clouds').add(img_cloud_shadow.select('shadows')).g
t(0)

    # Remove small cloud-shadow patches and dilate remaining pixels by
BUFFER input.
    # 20 m scale is for speed, and assumes clouds don't require 10 m
precision.
    is_cld_shdw = (is_cld_shdw.focalMin(2).focalMax(BUFFER*2/20)
        .reproject(**{'crs': img.select([0]).projection(), 'scale': 20})
        .rename('cloudmask'))

    # Add the final cloud-shadow mask to the image.
    return img_cloud_shadow.addBands(is_cld_shdw)
```

In [ ]:

```
def apply_cld_shdw_mask(img):
    # Subset the cloudmask band and invert it so clouds/shadow are 0, else
1.
    not_cld_shdw = img.select('cloudmask').Not()

    # Subset reflectance bands and update their masks, return the result.
    return img.select('B.*').updateMask(not_cld_shdw)

#----------------------------------FIN DU MASQUE NUAGEUX------------------
-----------------
```

In [ ]:

```
s2_sr = s2_sr_cld_col.map(add_cld_shdw_mask).map(apply_cld_shdw_mask)
```

In [ ]:

```
#changer date system pour mieux filtrer après, + clip

tonlecoll_new = s2_sr.map(lambda img: img.set({"DATE":
ee.Date(img.get("system:time_start"))\
                                        .format("YYYY-dd-
MM")}).clip(AOIlarge))
```

In [ ]:

```
#saison humide stats
```

```
def wet(year):
    start_date = ee.Date.fromYMD(year, 1, 1)
    end_date = start_date.advance(1, 'year')

    image = tonlecoll_new.filterDate(start_date, end_date)\
    .filterMetadata('DATE', 'not_ends_with', '-01')\
    .filterMetadata('DATE', 'not_ends_with', '-02')\
    .filterMetadata('DATE', 'not_ends_with', '-03')\
    .filterMetadata('DATE', 'not_ends_with', '-04')\
    .filterMetadata('DATE', 'not_ends_with', '-05')\
    .filterMetadata('DATE', 'not_ends_with', '-06')\
    .filterMetadata('DATE', 'not_ends_with', '-07')\
    .filterMetadata('DATE', 'not_ends_with', '-12')\
    .median()
    return image

#definir liste années
start_year = 2019
end_year = 2021
years = ee.List.sequence(start_year, end_year)
year_list = years.getInfo()

#assembler années & collection saison
images = years.map(wet)

#en faire une image collection
wetcoll = ee.ImageCollection(images)

#def fontion pour extraire ndwi
def extract_water(img):
    mndwi_threshold = 0
    mndwi_image = img.normalizedDifference(['B3', 'B11'])
    water_image = mndwi_image.gt(mndwi_threshold).selfMask()
    return water_image

#appliquer fonction extraction ndwi
mndwi_wet = wetcoll.map(extract_water)

#stats saison
def area_season(img):
    wclass = img.select('nd')
    areaImage = img.pixelArea().divide(1e6).addBands(wclass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': AOIlarge,
      'scale': 10,
      'maxPixels': 1e9
    })
    return img.set({'season_area': stats})

res_wet = mndwi_wet.map(area_season)

res_list = ee.List(res_wet.aggregate_array('season_area'))
```

```
finalzip = res_list.zip(year_list)

#definir liste SEASON
wet = ee.List(['wet', 'wet', 'wet'])
dry = ee.List(['dry', 'dry', 'dry'])

final = finalzip.zip(wet)
final.getInfo()
```

In [ ]:

```
#saison sèche stats

def dry(year):
    start_date = ee.Date.fromYMD(year, 1, 1)
    end_date = start_date.advance(1, 'year')

    image = tonlecoll_new.filterDate(start_date, end_date)\
    .filterMetadata('DATE', 'not_ends_with', '-01')\
    .filterMetadata('DATE', 'not_ends_with', '-06')\
    .filterMetadata('DATE', 'not_ends_with', '-07')\
    .filterMetadata('DATE', 'not_ends_with', '-08')\
    .filterMetadata('DATE', 'not_ends_with', '-09')\
    .filterMetadata('DATE', 'not_ends_with', '-10')\
    .filterMetadata('DATE', 'not_ends_with', '-11')\
    .filterMetadata('DATE', 'not_ends_with', '-12')\
    .median()
    return image

#definir liste années
start_year = 2019
end_year = 2021
years = ee.List.sequence(start_year, end_year)
year_list = years.getInfo()

#assembler années & collection saison
images = years.map(dry)

#en faire une image collection
drycoll = ee.ImageCollection(images)

#def fontion pour extraire ndwi
def extract_water(img):
    mndwi_threshold = 0
    mndwi_image = img.normalizedDifference(['B3', 'B11'])
    water_image = mndwi_image.gt(mndwi_threshold).selfMask()
    return water_image

#appliquer fonction extraction ndwi
mndwi_dry = drycoll.map(extract_water)

#stats saison
def area_season(img):
    wclass = img.select('nd')
    areaImage = img.pixelArea().divide(1e6).addBands(wclass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
```

```
      }),
      'geometry': AOIlarge,
      'scale': 10,
      'maxPixels': 1e9
    })
    return img.set({'season_area': stats})

res_dry = mndwi_dry.map(area_season)

res_list = ee.List(res_dry.aggregate_array('season_area'))

finalzip = res_list.zip(year_list)

#definir liste SEASON
wet = ee.List(['wet', 'wet', 'wet'])
dry = ee.List(['dry', 'dry', 'dry'])

final = finalzip.zip(dry)
final.getInfo()
```

In [ ]:

```
#ajouter layers à la Map

mndwi_wet_img21 = mndwi_wet.filterMetadata('system:index', 'equals',
'2').first()
Map.addLayer(mndwi_wet_img21, {'palette': '6495ED'}, 'mndwi_wet_img21',
False)

mndwi_dry_img21 = mndwi_dry.filterMetadata('system:index', 'equals',
'2').first()
Map.addLayer(mndwi_dry_img21, {'palette': '00FFFF'}, 'mndwi_dry_img21',
False)

legend_dict = {
    'Wet (3768,34 km2)': '6495ED',
    'Dry (2510,34 km2)': '00FFFF'}

Map.add_legend(legend_title="", legend_dict=legend_dict)
```

In [ ]:

```
#afficher la map

Map = geemap.Map()
Map.add_basemap('SATELLITE')
Map.setCenter(104.2, 12.8, 9)
Map
```

In [ ]:

```
#stats annuelles

def yearly(year):
    start_date = ee.Date.fromYMD(year, 1, 1)
    end_date = start_date.advance(1, 'year')

    image = tonlecoll_new.filterDate(start_date, end_date).median()
    return image
```

In [ ]:

```
#definir liste années
start_year = 2019
end_year = 2021
```

```
years = ee.List.sequence(start_year, end_year)
year_list = years.getInfo()

#assembler années & collection saison
images = years.map(yearly)

#en faire une image collection
yearlycoll = ee.ImageCollection(images)
```

In [ ]:

```
#def fontion pour extraire ndwi
def extract_water(img):
    mndwi_threshold = 0
    mndwi_image = img.normalizedDifference(['B3', 'B11'])
    water_image = mndwi_image.gt(mndwi_threshold).selfMask()
    return water_image

#appliquer fonction extraction ndwi
mndwi_yearly = yearlycoll.map(extract_water)
```

In [ ]:

```
#stats saison

def area_season(img):
    wclass = img.select('nd')
    areaImage = img.pixelArea().divide(1e6).addBands(wclass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': AOIlarge,
      'scale': 10,
      'maxPixels': 1e9
    })
    return img.set({'season_area': stats})

res_yearly = mndwi_yearly.map(area_season)
```

In [ ]:

```
res_list = ee.List(res_yearly.aggregate_array('season_area'))

final = res_list.zip(year_list)

final.getInfo()
```

In [ ]:

```
#ajouter layers a la map

mndwi_yearly_img19 = mndwi_yearly.filterMetadata('system:index', 'equals',
'0')
Map2.addLayer(mndwi_yearly_img19, {'palette': '6495ED'},
'mndwi_yearly_img19', False)

mndwi_yearly_img21 = mndwi_yearly.filterMetadata('system:index', 'equals',
'2')
Map2.addLayer(mndwi_yearly_img21, {'palette': '00FFFF'},
'mndwi_yearly_img21', False)

legend_dict = {
    '2019 (2775,96 km2)': '6495ED',
```

```
    '2021 (2574,40 km2)': '00FFFF'}

Map2.add_legend(legend_title="", legend_dict=legend_dict)
```

In [ ]:

```
#afficher la map

Map2 = geemap.Map()
Map2.add_basemap('SATELLITE')
Map2.setCenter(104.2, 12.8, 9)
Map2
```

```
#----------------------------------SCRIPT 5: SUESCA SENTINEL-------------
---------------------
```

```
import os
import geemap
import ee
```

In [ ]:

```
#initialiser la librairie

ee.Initialize()
```

In [ ]:

```
#geemap.update_package()
```

In [ ]:

```
#définir nouvelle roi et valeurs pour suesca

AOI = ee.Geometry.Polygon([[[-73.806998, 5.173556],
   [-73.798954, 5.151531],
   [-73.75932, 5.200934],
   [-73.77549, 5.209497],
   [-73.806998, 5.173556]]])
START_DATE = '2019'
END_DATE = '2022'
CLOUD_FILTER = 60
CLD_PRB_THRESH = 40
NIR_DRK_THRESH = 0.15
CLD_PRJ_DIST = 2
BUFFER = 100
```

In [ ]:

```
#----------------------------------DEBUT DU MASQUE NUAGEUX----------------
-------------------

def get_s2_sr_cld_col(aoi, start_date, end_date):
    # Import and filter S2 SR.
    s2_sr_col = (ee.ImageCollection('COPERNICUS/S2_SR')
        .filterBounds(aoi)
        .filterDate(start_date, end_date)
        .filter(ee.Filter.lte('CLOUDY_PIXEL_PERCENTAGE', CLOUD_FILTER)))

    # Import and filter s2cloudless.
    s2_cloudless_col =
(ee.ImageCollection('COPERNICUS/S2_CLOUD_PROBABILITY')
        .filterBounds(aoi)
        .filterDate(start_date, end_date))

    # Join the filtered s2cloudless collection to the SR collection by the
'system:index' property.
    return ee.ImageCollection(ee.Join.saveFirst('s2cloudless').apply(**{
        'primary': s2_sr_col,
        'secondary': s2_cloudless_col,
        'condition': ee.Filter.equals(**{
            'leftField': 'system:index',
            'rightField': 'system:index'
        })
    }))
```

In [ ]:

```
s2_sr_cld_col = get_s2_sr_cld_col(AOI, START_DATE, END_DATE)
```

In [ ]:

```
def add_cloud_bands(img):
    # Get s2cloudless image, subset the probability band.
    cld_prb = ee.Image(img.get('s2cloudless')).select('probability')

    # Condition s2cloudless by the probability threshold value.
    is_cloud = cld_prb.gt(CLD_PRB_THRESH).rename('clouds')

    # Add the cloud probability layer and cloud mask as image bands.
    return img.addBands(ee.Image([cld_prb, is_cloud]))
```

In [ ]:

```
def add_shadow_bands(img):
    # Identify water pixels from the SCL band.
    not_water = img.select('SCL').neq(6)

    # Identify dark NIR pixels that are not water (potential cloud shadow
pixels).
    SR_BAND_SCALE = 1e4
    dark_pixels =
img.select('B8').lt(NIR_DRK_THRESH*SR_BAND_SCALE).multiply(not_water).renam
e('dark_pixels')

    # Determine the direction to project cloud shadow from clouds (assumes
UTM projection).
    shadow_azimuth =
ee.Number(90).subtract(ee.Number(img.get('MEAN_SOLAR_AZIMUTH_ANGLE')));

    # Project shadows from clouds for the distance specified by the
CLD_PRJ_DIST input.
    cld_proj =
(img.select('clouds').directionalDistanceTransform(shadow_azimuth,
CLD_PRJ_DIST*10)
        .reproject(**{'crs': img.select(0).projection(), 'scale': 100})
        .select('distance')
        .mask()
        .rename('cloud_transform'))

    # Identify the intersection of dark pixels with cloud shadow
projection.
    shadows = cld_proj.multiply(dark_pixels).rename('shadows')

    # Add dark pixels, cloud projection, and identified shadows as image
bands.
    return img.addBands(ee.Image([dark_pixels, cld_proj, shadows]))
```

In [ ]:

```
def add_cld_shdw_mask(img):
    # Add cloud component bands.
    img_cloud = add_cloud_bands(img)

    # Add cloud shadow component bands.
    img_cloud_shadow = add_shadow_bands(img_cloud)

    # Combine cloud and shadow mask, set cloud and shadow as value 1, else
0.
    is_cld_shdw =
img_cloud_shadow.select('clouds').add(img_cloud_shadow.select('shadows')).g
t(0)
```

```
    # Remove small cloud-shadow patches and dilate remaining pixels by
BUFFER input.
    # 20 m scale is for speed, and assumes clouds don't require 10 m
precision.
    is_cld_shdw = (is_cld_shdw.focalMin(2).focalMax(BUFFER*2/20)
        .reproject(**{'crs': img.select([0]).projection(), 'scale': 20})
        .rename('cloudmask'))

    # Add the final cloud-shadow mask to the image.
    return img_cloud_shadow.addBands(is_cld_shdw)
```

In [ ]:

```
def apply_cld_shdw_mask(img):
    # Subset the cloudmask band and invert it so clouds/shadow are 0, else
1.
    not_cld_shdw = img.select('cloudmask').Not()

    # Subset reflectance bands and update their masks, return the result.
    return img.select('B.*').updateMask(not_cld_shdw)

#----------------------------------FIN DU MASQUE NUAGEUX------------------
-----------------
```

In [ ]:

```
s2_sr = s2_sr_cld_col.map(add_cld_shdw_mask).map(apply_cld_shdw_mask)
```

In [ ]:

```
#changer date system pour mieux filtrer après, + clip
suescacoll_new = s2_sr.map(lambda img: img.set({"DATE":
ee.Date(img.get("system:time_start"))\
                                        .format("YYYY-dd-MM")}).clip(AOI))
```

In [ ]:

```
#stats saison humide

def wet(year):
    start_date = ee.Date.fromYMD(year, 1, 1)
    end_date = start_date.advance(1, 'year')

    image = suescacoll_new.filterDate(start_date, end_date)\
    .filterMetadata('DATE', 'not_ends_with', '-01')\
    .filterMetadata('DATE', 'not_ends_with', '-02')\
    .filterMetadata('DATE', 'not_ends_with', '-03')\
    .filterMetadata('DATE', 'not_ends_with', '-04')\
    .filterMetadata('DATE', 'not_ends_with', '-05')\
    .filterMetadata('DATE', 'not_ends_with', '-06')\
    .filterMetadata('DATE', 'not_ends_with', '-07')\
    .filterMetadata('DATE', 'not_ends_with', '-08')\
    .filterMetadata('DATE', 'not_ends_with', '-09')\
    .mean()
    return image

#definir liste années
start_year = 2019
end_year = 2021
years = ee.List.sequence(start_year, end_year)
year_list = years.getInfo()

#assembler années & collection saison
images = years.map(wet)
```

```
#en faire une image collection
wetcoll = ee.ImageCollection(images)

#def fontion pour extraire ndwi
def extract_water(img):
    mndwi_threshold = 0
    mndwi_image = img.normalizedDifference(['B3', 'B11'])
    water_image = mndwi_image.gt(mndwi_threshold).selfMask()
    return water_image

#appliquer fonction extraction ndwi
mndwi_wet = wetcoll.map(extract_water)

#stats saison
def area_season(img):
    wclass = img.select('nd')
    areaImage = img.pixelArea().divide(1e6).addBands(wclass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': AOI,
      'scale': 10,
      'maxPixels': 1e9
    })
    return img.set({'season_area': stats})

res_wet = mndwi_wet.map(area_season)

res_list = ee.List(res_wet.aggregate_array('season_area'))

finalzip = res_list.zip(year_list)

#definir liste SEASON
wet = ee.List(['wet', 'wet', 'wet'])
dry = ee.List(['dry', 'dry', 'dry'])

final = finalzip.zip(wet)
final.getInfo()
```

In [ ]:

```
#stats saison sèche

def dry(year):
    start_date = ee.Date.fromYMD(year, 1, 1)
    end_date = start_date.advance(1, 'year')

    image = suescacoll_new.filterDate(start_date, end_date)\
    .filterMetadata('DATE', 'not_ends_with', '-03')\
    .filterMetadata('DATE', 'not_ends_with', '-04')\
    .filterMetadata('DATE', 'not_ends_with', '-05')\
    .filterMetadata('DATE', 'not_ends_with', '-06')\
    .filterMetadata('DATE', 'not_ends_with', '-07')\
    .filterMetadata('DATE', 'not_ends_with', '-08')\
    .filterMetadata('DATE', 'not_ends_with', '-09')\
    .filterMetadata('DATE', 'not_ends_with', '-10')\
    .filterMetadata('DATE', 'not_ends_with', '-11')\
```

```
    .filterMetadata('DATE', 'not_ends_with', '-12')\
    .median()
    return image

#definir liste années
start_year = 2019
end_year = 2021
years = ee.List.sequence(start_year, end_year)
year_list = years.getInfo()

#assembler années & collection saison
images = years.map(dry)

#en faire une image collection
drycoll = ee.ImageCollection(images)

#def fontion pour extraire ndwi
def extract_water(img):
    mndwi_threshold = 0
    mndwi_image = img.normalizedDifference(['B3', 'B11'])
    water_image = mndwi_image.gt(mndwi_threshold).selfMask()
    return water_image

#appliquer fonction extraction ndwi
mndwi_dry = drycoll.map(extract_water)

#stats saison
def area_season(img):
    wclass = img.select('nd')
    areaImage = img.pixelArea().divide(1e6).addBands(wclass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': AOI,
      'scale': 10,
      'maxPixels': 1e9
    })
    return img.set({'season_area': stats})

res_dry = mndwi_dry.map(area_season)

res_list = ee.List(res_dry.aggregate_array('season_area'))

finalzip = res_list.zip(year_list)

#definir liste SEASON
wet = ee.List(['wet', 'wet', 'wet'])
dry = ee.List(['dry', 'dry', 'dry'])

final = finalzip.zip(dry)
final.getInfo()
```

In [ ]:

```
#ajouter layers à la Map

rgb = s2_sr.mean()
```

```
Map.addLayer(rgb, {'bands': ['B4', 'B3', 'B2'], 'min': 0, 'max': 2500,
'gamma': 1.1},
               'rgb_wet21', False, 1)

mndwi_wet_img21 = mndwi_wet.filterMetadata('system:index', 'equals', '2')
Map.addLayer(mndwi_wet_img21, {'palette': '6495ED'}, 'mndwi_wet_img21',
False)

mndwi_dry_img21 = mndwi_dry.filterMetadata('system:index', 'equals', '2')
Map.addLayer(mndwi_dry_img21, {'palette': '00FFFF'}, 'mndwi_dry_img21',
False)

legend_dict = {
    'Wet (1,45 km2)': '6495ED',
    'Dry (0,87 km2)': '00FFFF'}

Map.add_legend(legend_title="", legend_dict=legend_dict)
```

In [ ]:

```
#afficher la map

Map = geemap.Map()
Map.add_basemap(basemap='Esri.WorldShadedRelief')
Map.setCenter(-73.78, 5.19, 13)
Map
```

In [ ]:

```
#stats annuelles

def yearly(year):
    start_date = ee.Date.fromYMD(year, 1, 1)
    end_date = start_date.advance(1, 'year')

    image = suescacoll_new.filterDate(start_date, end_date).median()
    return image

#definir liste années
start_year = 2019
end_year = 2021
years = ee.List.sequence(start_year, end_year)
year_list = years.getInfo()

#assembler années & collection saison
images = years.map(yearly)

#en faire une image collection
yearlycoll = ee.ImageCollection(images)

#def fontion pour extraire ndwi
def extract_water(img):
    mndwi_threshold = 0
    mndwi_image = img.normalizedDifference(['B3', 'B11'])
    water_image = mndwi_image.gt(mndwi_threshold).selfMask()
    return water_image

#appliquer fonction extraction ndwi
mndwi_yearly = yearlycoll.map(extract_water)

#stats saison
```

```
def area_season(img):
    wclass = img.select('nd')
    areaImage = img.pixelArea().divide(1e6).addBands(wclass)
    stats = areaImage.reduceRegion(**{
      'reducer': ee.Reducer.sum().group(**{
        'groupField': 1,
        'groupName': 'waterclass_value',
      }),
      'geometry': AOI,
      'scale': 10,
      'maxPixels': 1e9
    })
    return img.set({'season_area': stats})

res_yearly = mndwi_yearly.map(area_season)

res_list = ee.List(res_yearly.aggregate_array('season_area'))

final = res_list.zip(year_list)


final.getInfo()
```

In [ ]:

```
#ajouter layers a la map

rgb = s2_sr.mean()
Map2.addLayer(rgb, {'bands': ['B4', 'B3', 'B2'], 'min': 0, 'max': 2500,
'gamma': 1.1},
                'rgb_wet21', False, 1)

mndwi_yearly_img19 = mndwi_yearly.filterMetadata('system:index', 'equals',
'0')
Map2.addLayer(mndwi_yearly_img19, {'palette': '6495ED'},
'mndwi_yearly_img19', False)

mndwi_yearly_img21 = mndwi_yearly.filterMetadata('system:index', 'equals',
'2')
Map2.addLayer(mndwi_yearly_img21, {'palette': '00FFFF'},
'mndwi_yearly_img21', False)

legend_dict = {
    '2019 (2,05 km2)': '6495ED',
    '2021 (0,89 km2)': '00FFFF'}

Map2.add_legend(legend_title="", legend_dict=legend_dict)
```

In [ ]:

```
#afficher la map

Map2 = geemap.Map()
Map2.add_basemap(basemap='Esri.WorldShadedRelief')
Map2.setCenter(-73.78, 5.19, 13)
Map2
```

```
#----------------------------------SCRIPT 6: WEB APP---------------------
-------------
```

```
import geemap
import ee
```

In [ ]:

```
#ajouter la map

Map = geemap.Map(center=[0.6175, 36.0747, 11], zoom=11)
```

In [ ]:

```
#initialiser la librairie

ee.Initialize()
```

In [ ]:

```
#définir les ROI et valeurs pour le masque nuageux

AOIba = ee.Geometry.Polygon([[[35.972779, 0.494408],
   [35.972779, 0.754325],
   [36.182682, 0.754325],
   [36.182682, 0.494408],
   [35.972779, 0.494408]]])

AOIto = ee.Geometry.Polygon([[[103.552867, 12.292278],
   [103.552867, 13.331446],
   [104.734406, 13.331446],
   [104.734406, 12.292278],
   [103.552867, 12.292278]]])


AOIsu = ee.Geometry.Polygon([[[-73.819219, 5.157686],
   [-73.819219, 5.218441],
   [-73.745784, 5.218441],
   [-73.745784, 5.157686],
   [-73.819219, 5.157686]]])

START_DATE = '2019'
END_DATE = '2022'
CLOUD_FILTER = 60
CLD_PRB_THRESH = 40
NIR_DRK_THRESH = 0.15
CLD_PRJ_DIST = 2
BUFFER = 100
```

In [ ]:

```
#----------------------------------DEBUT DU MASQUE NUAGEUX----------------
------------------

def get_s2_sr_cld_col(aoi, start_date, end_date):
    # Import and filter S2 SR.
    s2_sr_col = (ee.ImageCollection('COPERNICUS/S2_SR')
        .filterBounds(aoi)
        .filterDate(start_date, end_date)
        .filter(ee.Filter.lte('CLOUDY_PIXEL_PERCENTAGE', CLOUD_FILTER)))

    # Import and filter s2cloudless.
    s2_cloudless_col =
(ee.ImageCollection('COPERNICUS/S2_CLOUD_PROBABILITY')
        .filterBounds(aoi)
```

```
        .filterDate(start_date, end_date))

    # Join the filtered s2cloudless collection to the SR collection by the
'system:index' property.
    return ee.ImageCollection(ee.Join.saveFirst('s2cloudless').apply(**{
        'primary': s2_sr_col,
        'secondary': s2_cloudless_col,
        'condition': ee.Filter.equals(**{
            'leftField': 'system:index',
            'rightField': 'system:index'
        })
    }))
```

In [ ]:

```
s2_sr_cld_col_ba = get_s2_sr_cld_col(AOIba, START_DATE, END_DATE)
```

In [ ]:

```
s2_sr_cld_col_to = get_s2_sr_cld_col(AOIto, START_DATE, END_DATE)
```

In [ ]:

```
s2_sr_cld_col_su = get_s2_sr_cld_col(AOIsu, START_DATE, END_DATE)
```

In [ ]:

```
def add_cloud_bands(img):
    # Get s2cloudless image, subset the probability band.
    cld_prb = ee.Image(img.get('s2cloudless')).select('probability')

    # Condition s2cloudless by the probability threshold value.
    is_cloud = cld_prb.gt(CLD_PRB_THRESH).rename('clouds')

    # Add the cloud probability layer and cloud mask as image bands.
    return img.addBands(ee.Image([cld_prb, is_cloud]))
```

In [ ]:

```
def add_shadow_bands(img):
    # Identify water pixels from the SCL band.
    not_water = img.select('SCL').neq(6)

    # Identify dark NIR pixels that are not water (potential cloud shadow
pixels).
    SR_BAND_SCALE = 1e4
    dark_pixels =
img.select('B8').lt(NIR_DRK_THRESH*SR_BAND_SCALE).multiply(not_water).renam
e('dark_pixels')

    # Determine the direction to project cloud shadow from clouds (assumes
UTM projection).
    shadow_azimuth =
ee.Number(90).subtract(ee.Number(img.get('MEAN_SOLAR_AZIMUTH_ANGLE')));

    # Project shadows from clouds for the distance specified by the
CLD_PRJ_DIST input.
    cld_proj =
(img.select('clouds').directionalDistanceTransform(shadow_azimuth,
CLD_PRJ_DIST*10)
        .reproject(**{'crs': img.select(0).projection(), 'scale': 100})
        .select('distance')
        .mask()
        .rename('cloud_transform'))

    # Identify the intersection of dark pixels with cloud shadow
projection.
```

```
    shadows = cld_proj.multiply(dark_pixels).rename('shadows')

    # Add dark pixels, cloud projection, and identified shadows as image
bands.
    return img.addBands(ee.Image([dark_pixels, cld_proj, shadows]))
```

In [ ]:

```
def add_cld_shdw_mask(img):
    # Add cloud component bands.
    img_cloud = add_cloud_bands(img)

    # Add cloud shadow component bands.
    img_cloud_shadow = add_shadow_bands(img_cloud)

    # Combine cloud and shadow mask, set cloud and shadow as value 1, else
0.
    is_cld_shdw =
img_cloud_shadow.select('clouds').add(img_cloud_shadow.select('shadows')).g
t(0)

    # Remove small cloud-shadow patches and dilate remaining pixels by
BUFFER input.
    # 20 m scale is for speed, and assumes clouds don't require 10 m
precision.
    is_cld_shdw = (is_cld_shdw.focalMin(2).focalMax(BUFFER*2/20)
        .reproject(**{'crs': img.select([0]).projection(), 'scale': 20})
        .rename('cloudmask'))

    # Add the final cloud-shadow mask to the image.
    return img_cloud_shadow.addBands(is_cld_shdw)
```

In [ ]:

```
def apply_cld_shdw_mask(img):
    # Subset the cloudmask band and invert it so clouds/shadow are 0, else
1.
    not_cld_shdw = img.select('cloudmask').Not()

    # Subset reflectance bands and update their masks, return the result.
    return img.select('B.*').updateMask(not_cld_shdw)

#----------------------------------FIN DU MASQUE NUAGEUX-----------------
-----------------
```

In [ ]:

```
s2_sr_ba = s2_sr_cld_col_ba.map(add_cld_shdw_mask).map(apply_cld_shdw_mask)
```

In [ ]:

```
s2_sr_to = s2_sr_cld_col_to.map(add_cld_shdw_mask).map(apply_cld_shdw_mask)
```

In [ ]:

```
s2_sr_su = s2_sr_cld_col_su.map(add_cld_shdw_mask).map(apply_cld_shdw_mask)
```

In [ ]:

```
#definir les images à comparer
```

In [ ]:

```
baringo2019gauche = s2_sr_ba.filterDate('2019', '2020').map(lambda img:
img.clip(AOIba)).median()
```

In [ ]:

```
baringo2021droite = s2_sr_ba.filterDate('2021', '2022').map(lambda img:
img.clip(AOIba)).median()
```

In [ ]:

```
tonle2019gauche = s2_sr_to.filterDate('2019', '2020').map(lambda img:
img.clip(AOIto)).median()
```
In [ ]:
```
tonle2021droite = s2_sr_to.filterDate('2021', '2022').map(lambda img:
img.clip(AOIto)).median()
```
In [ ]:
```
suesca2019gauche = s2_sr_su.filterDate('2019', '2020').map(lambda img:
img.clip(AOIsu)).median()
```
In [ ]:
```
suesca2021droite = s2_sr_su.filterDate('2021', '2022').map(lambda img:
img.clip(AOIsu)).median()
```
In [ ]:
```
#definir les parametres visuels pour composition colorée

vis_params_true = {
  'bands': ['B4', 'B3', 'B2'],
  'min': 0,
  'max': 2500,
  'gamma': 1.1,
}
```
In [ ]:
```
#définir les listes pour les menus déroulants
```
In [ ]:
```
gauche2019 = ee.ImageCollection([baringo2019gauche, tonle2019gauche,
suesca2019gauche])
```
In [ ]:
```
droite2021 = ee.ImageCollection([baringo2021droite, tonle2021droite,
suesca2021droite])
```
In [ ]:
```
gauche_layer_names = ['Baringo 2019 (190,20 km2)', 'Tonlé Sap 2019 (2775,96
km2)', 'Laguna de Suesca 2019 (2,05 km2)']
```
In [ ]:
```
droite_layer_names = ['Baringo 2021 (228,03 km2)', 'Tonlé Sap 2021 (2574,40
km2)', 'Laguna de Suesca 2021 (0,89 km2)']
```
In [ ]:
```
#ajouter le ts_inspector pour séparer la map

Map.ts_inspector(left_ts=gauche2019, right_ts=droite2021,
left_names=gauche_layer_names, right_names=droite_layer_names,
                 left_vis=vis_params_true, right_vis=vis_params_true)
```
In [ ]:
```
#afficher la map

Map
```

10. LEXIQUE

GEE : Google Earth Engine

HSV : Hue saturation value, la value de la saturation de la teinte

MNDWI : Modified Normalized Differenciation Water Index, l'index modifié de mesure de l'eau

NDVI : Normalized Differenced Vegetation Index, l'index de mesure de la vegetation

NDWI : Normalized Differenced Water Index, l'index de mesure de l'eau

NIR : Near Infra Red, la bande proche Infra-rouge

SIG : Système d'Information Géographique

SWIR1 : Short Wave Infra Red 1, la bande à courte longeur d'onde Infra-rouge

WRS_PATH : La colonne du World Wide Reference System pour le satellite Landsat

WRS_ROW : Le rang du World Wide Reference System pour le satellite Landsat