

Identification et Mise à jour de l'Infrastructure Écologique

Guide utilisateur - Version française

Audrey Lambiel

2025-10-15

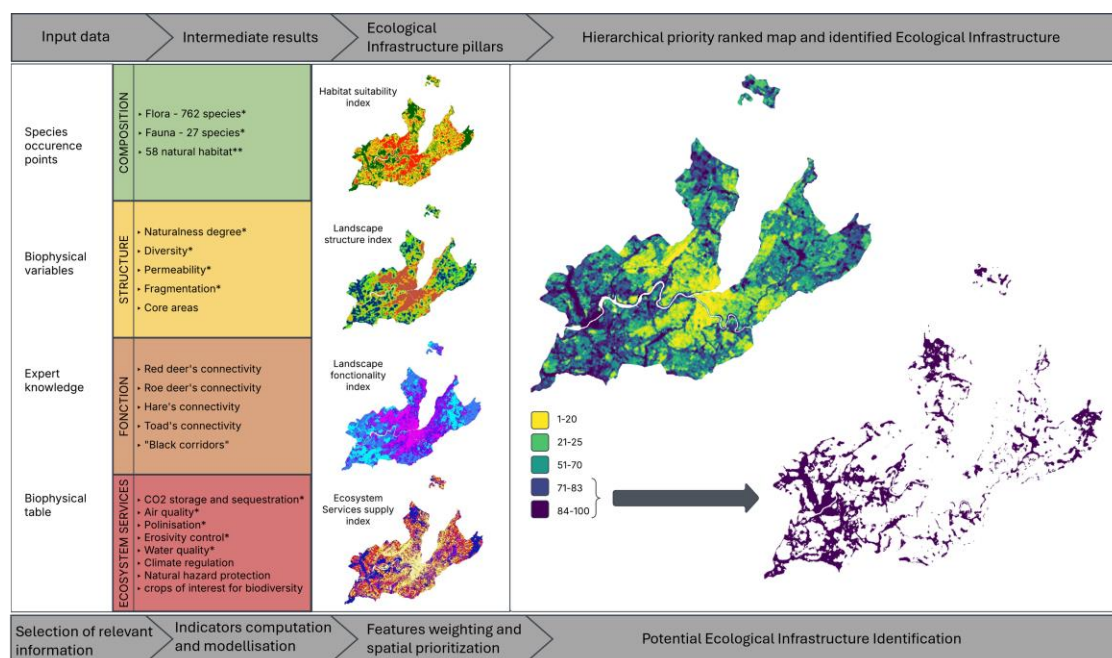
Introduction	2
Objectifs de ce guide pratique	3
Qu'est-ce que JupyterLab ? Une rapide introduction	3
Configuration requise du système.....	6
Étape 1 : mettre en place l'espace de travail	7
Architecture de l'environnement.....	7
Cloner le répertoire GitHub.....	7
Déployer les environnements virtuels.....	8
Installer de Zonation 5	9
Accéder à votre base de données partagée	10
Étape 2 : exécution des scripts.....	11
Indicateurs	11
Priorisation	16
Résultats et métadonnées.....	17
Ressources	18
Lexique	18
Quelques commandes utiles	19
Références et liens utiles.....	21

Introduction

La planification environnementale et la conservation de la biodiversité reposent de plus en plus sur des infrastructures numériques reproductibles, transparentes et évolutives. Dans ce contexte, **JupyterLab** constitue un environnement de développement interactif particulièrement adapté à la gestion de workflows complexes, notamment ceux impliquant de grandes quantités de données spatiales et écologiques.

Ce guide accompagne la note technique intitulée « *A Digital Twin approach for the identification and update of Ecological Infrastructure* » et fournit des instructions pratiques pour reproduire les analyses présentées dans l'article. Il a été conçu pour permettre aux utilisateur.trice.s de déployer l'environnement partagé JupyterLab, d'installer les composants logiciels nécessaires, et de mettre en œuvre la méthode appliquée pour identifier et mettre à jour l'Infrastructure Écologique (IE) du canton de Genève.

La méthodologie repose sur un **diagnostic de biodiversité** structuré autour de quatre piliers thématiques, chacun composé d'indicateurs écologiques. Ces indicateurs sont calculés via une chaîne de traitement automatisée en **langage R**, et les résultats sont hiérarchisés à l'aide de l'outil de planification spatiale **Zonation 5**. Tous les scripts sont annotés et la génération automatique de métadonnées garantit la traçabilité et la reproductibilité des résultats.



La figure ci-dessus présente le cadre général utilisé pour établir L'IE dans le canton de Genève sur la base des quatre piliers de la biodiversité selon Honeck, Sanguet, et al., (2020). Après avoir classé hiérarchiquement tous les pixels entre 1 et 100, seuls les 30 % les meilleurs sont conservés, puis identifiés comme infrastructure écologique. Dans la figure, les indicateurs marqués par un * sont ceux actuellement sélectionnés et pleinement mis en œuvre dans le processus d'automatisation. Ceux marqués par ** sont actuellement sélectionnés et partiellement mis en œuvre (c'est-à-dire que l'automatisation a été réalisée

pour tous les habitats naturels, mais pas pour la couche des arbres isolés) dans le processus d'automatisation.

Le matériel présent dans le répertoire [GitHub](#) contient des scripts R, des fichiers de configuration et de la documentation pour l'identification et la mise à jour de l'infrastructure écologique (IE), conformément à la méthode proposée par [GE21](#). Ce travail a été conçu pour être utilisé via un environnement partagé sur JupyterHub et les scripts ont donc été écrits pour être exécutés dans ce type d'environnement.

En suivant ce guide, les utilisateur.trice.s pourront reproduire les analyses en adaptant la méthode à d'autres territoires ou jeux de données.

Objectifs de ce guide pratique

Ce guide a pour objectif de fournir aux utilisateur.trice.s une documentation claire et opérationnelle pour :

- **Prendre en main** JupyterLab
- **Installer les environnements logiciels** nécessaires à l'exécution des scripts
- **Accéder aux scripts** disponibles dans le dépôt GitHub du projet
- **Exécuter les scripts** permettant le calcul des indicateurs et la priorisation spatiale
- **Comprendre la structure des résultats** produits, ainsi que les métadonnées associées

Il s'adresse à toute personne souhaitant reproduire les analyses présentées dans la publication, adapter la méthode à d'autres territoires, ou contribuer au développement d'infrastructures numériques FAIR pour la planification environnementale.

Qu'est-ce que JupyterLab ? Une rapide introduction

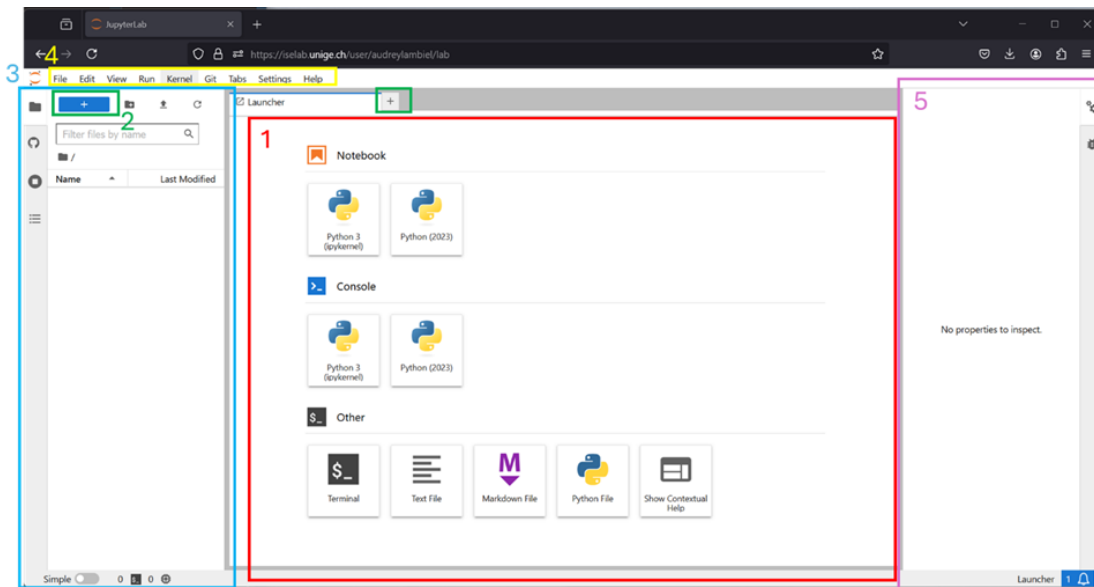
JupyterLab est un environnement de développement interactif, open source, conçu pour la recherche scientifique et l'analyse de données. Il permet d'écrire, d'exécuter et de documenter du code dans plusieurs langages (notamment R et Python), tout en intégrant des visualisations, des fichiers texte, des consoles et des terminaux dans une interface unifiée.

Dans le cadre de ce projet, JupyterLab est utilisé comme **infrastructure centrale** pour :

- Automatiser le calcul des indicateurs écologiques
- Centraliser les données et les scripts
- Faciliter le travail collaboratif entre partenaires
- Assurer la traçabilité et la reproductibilité des résultats

L'environnement est déployé sur un serveur partagé (JupyterHub), mais peut également être installé localement. Il permet à chaque utilisateur.trice.s de travailler dans un espace personnel tout en accédant à des ressources communes (scripts, données, environnements virtuels).

Interface JupyterLab



L'interface de JupyterLab comprend 5 principaux éléments :

1. **Une zone de travail principale** avec les onglets de documents et des activités. Vous pouvez dans cette zone organiser des documents (fichiers texte, notebook, etc.) et d'autres activités comme des terminaux ou des consoles de code en panneaux d'onglets redimensionnables et subdivisables (faites un 'drag and drop' d'un nouvel onglet sur le panneau souhaitez). Dans la capture d'écran ci-après, nous avons organisé dans notre espace de travail principal en trois blocs différents, avec en haut à gauche un notebook, en haut à droite une nouvelle page de Launcher et la partie inférieure de l'écran, un terminal.
2. **Un onglet launcher** qui permet d'afficher les kernels ainsi que les consoles disponibles. Il est donc possible de démarrer un nouveau notebook/console basés sur n'importe lequel d'entre eux. Il permet également de lancer un nouveau terminal.
3. **Une barre latérale gauche** avec notamment :
 1. **Le navigateur de fichier** où les icônes supérieures vous permettent de lancer un nouveau Launcher, créer un nouveau dossier, télécharger un fichier, etc. Lorsque vous naviguez dans votre arborescence de dossier, les actions de créations de dossiers/fichiers se font automatiquement dans le dossier dans lequel vous êtes.

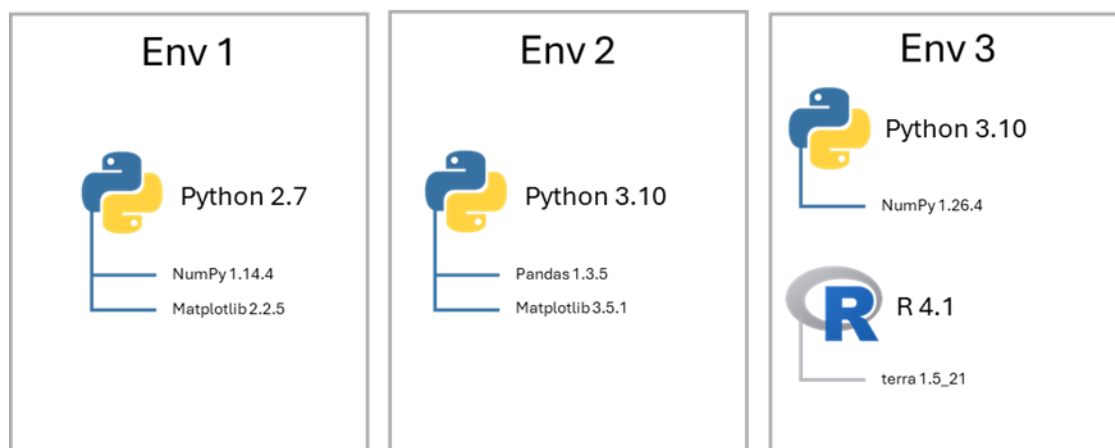
2. **La liste des kernels et des terminaux actifs ainsi que des onglets ouverts.** En effet, lorsque vous fermez un notebook, une console de code ou un terminal, le kernel sous-jacent ou le terminal fonctionnant sur le serveur continue de fonctionner. Cet onglet permet donc de voir quel kernels/terminaux sont encore en train de produire des calculs et de les rouvrir/fermer si besoin.
4. **Une barre de menu supérieure** permettant d'agir par exemple sur les fichiers actifs ou sur les kernels, l'affichage dans la zone de travail, etc.
5. **Une barre latérale droite** avec un inspecteur de propriété (actif dans les notebooks) et le débbugger.

Terminal

Les terminaux JupyterLab offrent une prise en charge complète des interpréteurs de commandes (bash, tsch, etc.) sur Mac/Linux et de PowerShell sur Windows. Les terminaux s'exécutent sur le système où tourne le serveur Jupyter. C'est également depuis le terminal que vous pouvez créer et gérer vos environnements virtuels. Pour ouvrir un nouveau terminal, cliquez sur + pour ouvrir un nouveau launcher et cliquez sur l'icône terminal. La fermeture d'un onglet de terminal le laisse en cours d'exécution sur le serveur, mais vous pouvez le rouvrir à l'aide de l'onglet Running terminals and kernels dans la barre latérale gauche.

Environnements virtuels

Un environnement virtuel est une instance isolée de l'interpréteur Python qui permet d'installer des bibliothèques et des dépendances spécifiques à un projet sans interférer avec d'autres projets ou avec les bibliothèques installées globalement sur le serveur. Lorsque vous travaillez avec JupyterLab, vous utilisez un kernel spécifique (par exemple Python) pour exécuter du code. Vous pouvez avoir plusieurs kernels et environnements virtuels pour différents projets, assurant ainsi que chaque projet utilise les bonnes versions des bibliothèques nécessaires. De même, il est possible de spécifier des variables d'environnement spécifiques à vos besoins.



Kernels

Les kernels (ou noyau en français) sont des processus distincts lancés par le serveur qui exécutent votre code dans différents langages et environnements de programmation personnalisés. JupyterLab vous permet de connecter n'importe quel fichier texte ouvert à une console de code et à un kernel. Ainsi, dans un environnement virtuel dans lequel Python et R seraient installés, on peut installer un kernel Python, permettant d'exécuter des lignes de code en Python, et un autre kernel R qui lui nous permettrait d'exécuter les lignes de commande R.

Les Kernels sont créés depuis un environnement virtuel activé via une commande simple dans le terminal. Une fois créé, le kernel est ensuite visible et peut être utilisé dans les notebooks et consoles.

Notebook

Les notebooks (.ipynb) sont des documents qui combinent du code directement exécutable et du texte narratif (Markdown) ou des équations (LaTeX), des images, etc..

Console

Les consoles de code vous permettent d'exécuter du code de manière interactive dans un kernel. Chaque ligne envoyée dans la console est immédiatement exécutée, et les variables mises en mémoire sont conservées.

Configuration requise du système

JupyterLab est multiplateforme, il fonctionne sous :

Windows	10 ou version ultérieure (64 bits)
macOS	10.12 (Sierra) ou version ultérieure
Linux	Ubuntu, Debian, Fedora, CentOS, Arch, etc.

Vous aurez besoin de Python (version 3.8 ou ultérieure) et pip (version ≥ 21.0 recommandée).

Le système peut être utilisé via un **serveur JupyterHub** (donc via un moteur de recherche, excepté Internet Explorer) ou **installé localement**. Dans ce cas, téléchargez JupyterLab Desktop ou utilisez exécutez dans le terminal de votre machine les lignes ci-dessous.

- avec pip

```
pip install jupyterlab
```

- ou via conda

```
conda install -c conda-forge jupyterlab
```

Étape 1 : mettre en place l'espace de travail

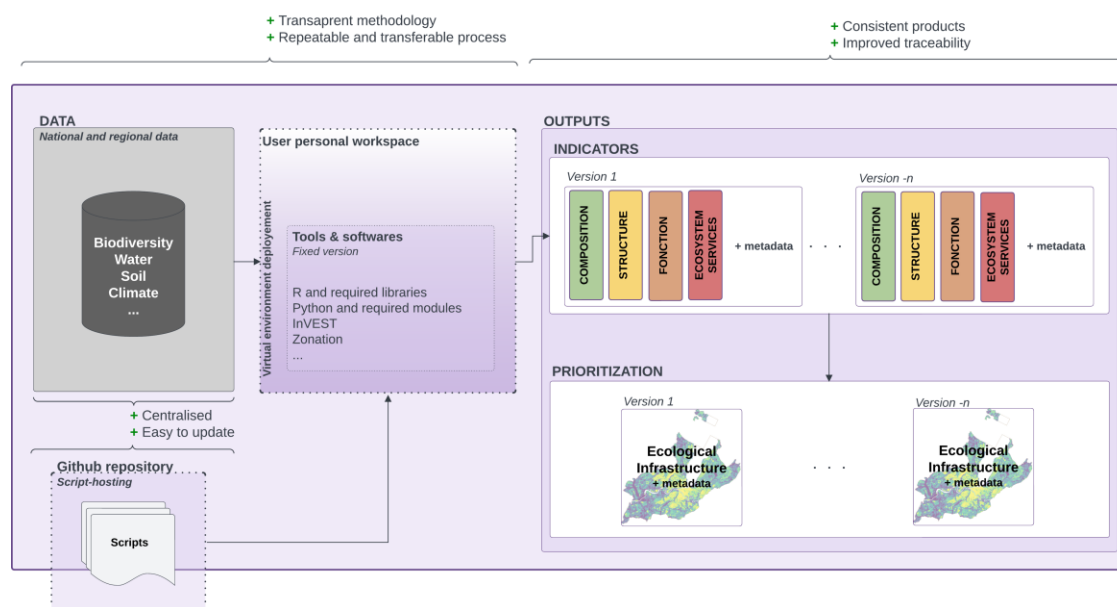
Architecture de l'environnement

Nous proposons de structurer l'environnement de travail autour de quatre composantes principales :

- Le **répertoire GitHub**, sur lequel sont stockés les scripts et les fichiers de configuration
- L'**espace personnel de l'utilisateur.trice** dans lequel les environnements sont déployés et les scripts exécutés.
- Un dossier DATA contenant les **données** nécessaires aux calculs des indicateurs.
- Le dossier OUTPUTS dans lequel les **résultats produits** sont sauvegardés automatiquement.

Dans la suite des explications de ce guide, nous partons du principe qu'un espace de travail partagé `shared_workspace` sur JupyterLab a été mis en place, et que celui-ci contient le dossier DATA. Il est cependant possible de travailler avec un dossier DATA directement dans son espace personnel.

N.B. : Une description plus détaillée de cette architecture est disponible dans la note technique associée (voir en fin de guide).



Cloner le répertoire GitHub

Depuis votre espace de travail `/home` sur JupyterLab, ouvrez un **terminal** et clonez le répertoire GitHub dans votre espace

```
git clone https://github.com/ALambiel/enviospace_IE
```

Le répertoire présente la structure suivante :

```
enviospace_IE/
├── config/                                # contient les fichiers environnements .yaml
│   ├── rspatial.yaml                    # analyses spatiales avec R
│   └── invest3141.yaml                  # InVEST avec python
├── run/
│   ├── <ind_pillar_indicator.r>         # calcul des indicateurs (un par
indicateur)
│   ├── template.r                      # un modèle pour créer d'autres indicateurs
│   ├── prio_zonation.r                 # exécuter la priorisation
│   └── <tools.r>                       # scripts additionnels pour prétraitement
└── README.md
```

Définissez ensuite le répertoire comme votre dossier de travail

```
cd enviospace_IE
```

Déployer les environnements virtuels

Les environnements virtuels sont définis dans deux fichiers de configuration :

- `rspatial.yaml` : pour les analyses spatiales en R
- `invest3141.yaml` : pour les indicateurs basés sur le modèle InVEST (R + Python)

Commencez ensuite par déployer l'environnement `rspatial` et créer le kernel associé

```
# créer l'environnement à partir du fichier .yaml
conda env create -f config/rspatial.yaml

# activer l'environnement
conda activate rspatial

# activez R
R

# installer le kernel
IRkernel::installspec(name = 'rspatial', displayname = 'Spatial analysis with
R')

# quittez R
q()
```

Faites de même pour l'environnement `invest3141`

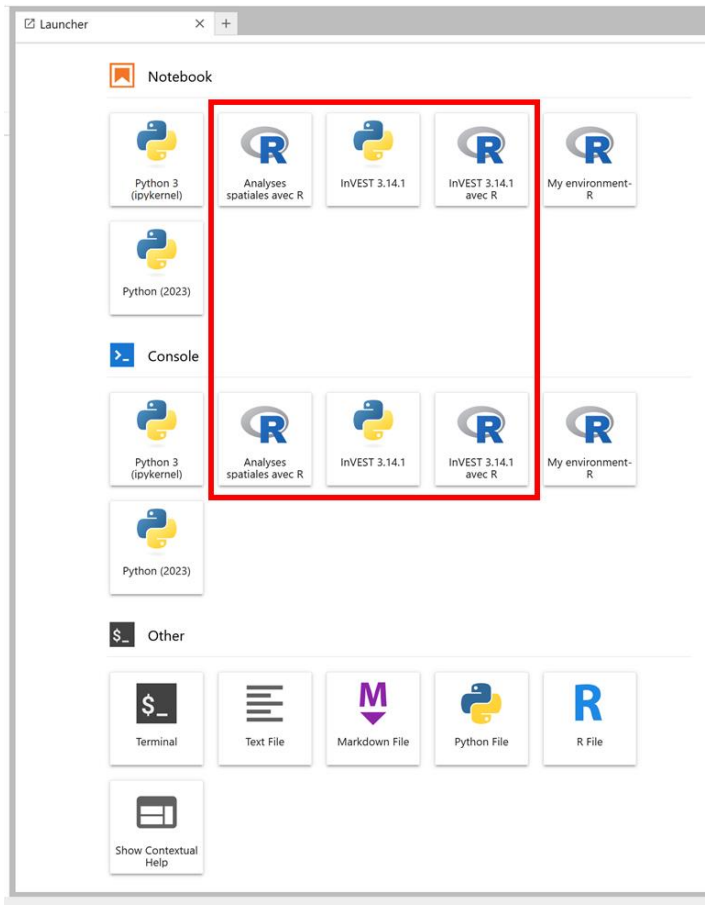
```
conda env create -f config/invest3141.yaml
conda activate invest3141
pip install natcap.invest==3.14.1
R
```



```
IRkernel::installspec(name = 'invest3141r', displayname = 'InVEST 3.14.1 with R')
q()

# installer aussi un kernel python
python -m ipykernel install --user --name invest3141 --display-name "InVEST 3.14.1"
```

Dans votre Launcher, les nouvelles icônes correspondantes aux environnements et kernels que vous venez de créer doivent apparaître.



Installer de Zonation 5

L'étape de priorisation (`prio_zonation.r`) repose sur l'utilisation du logiciel Zonation 5, qui doit être installé sous forme d'AppImage dans l'environnement JupyterLab.

Dans un premier temps télécharger l'AppImage de Zonation 5 en cliquant sur le lien suivant : [Zonation5_Linux.zip \(latest release\)](#).

Une fois le fichier zip téléchargé, dézippez-le et téléchargez le fichier `zonation5` dans un dossier sur votre environnement JupyterLab (par exemple dans `enviropspace_IE/zonation_folder`).

Enfin, rendez-le exécutable et extrayez l'AppImage en ouvrant un terminal et en exécutant les commandes suivantes :

```
cd /path/to/zonation_folder
chmod +x zonation5
./zonation5 --appimage-extract
```

Un dossier squashfs-root/ est maintenant présentant dans le dossier zonation_folder et il contient l'exécutable de Zonation.

Accéder à votre base de données partagée

N.B.: Comme mentionné précédemment, nous proposons de travailler avec une base de données partagée. Cela implique qu'un environnement partagé sur JupyterHub ai été mis en place et que votre administrateur.trice vous ai donné les accès à ce dossier et son contenu.

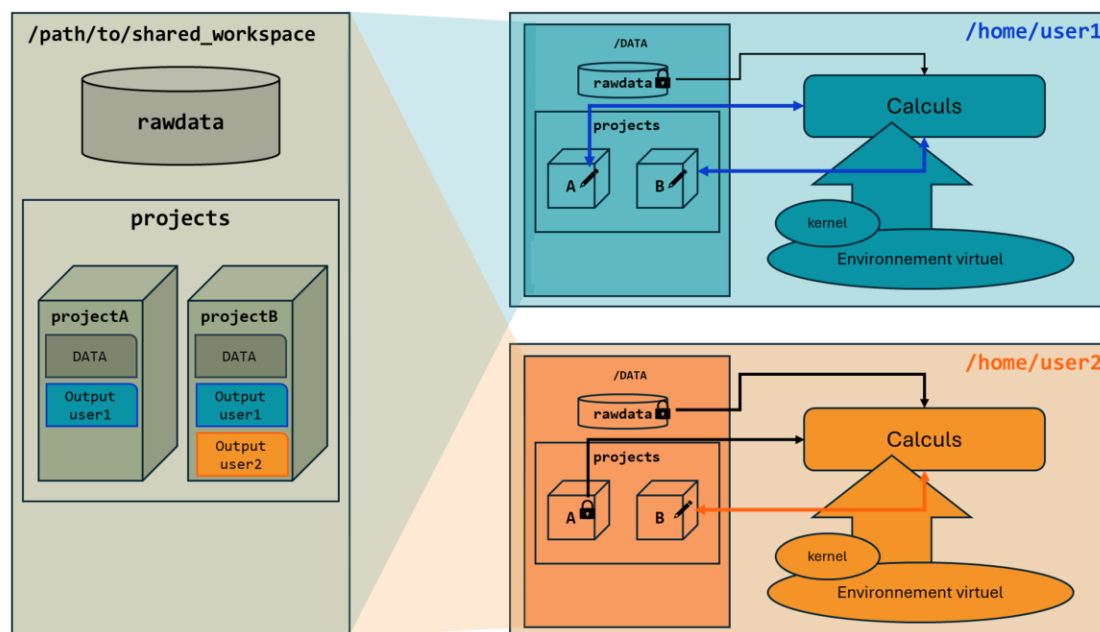
Dans l'explorateur de fichier, rendez-vous dans enviroSPACE_IE et ouvrez un terminal depuis le Launcher. Ensuite, exécuter :

```
ln -s /path/to/shared_workspace shared_env
```

Cela permet de créer un raccourci (ou une image) dans votre espace personnel vers shared_workspace: un dossier shared_env est maintenant présent à la racine de votre espace personnel. Vous pouvez interagir comme avec tous vos autres dossiers personnels.

Le dossier shared_env n'est pas physiquement présent dans votre espace personnel car en créant un lien symbolique enviroSPACE, vous avez créé un raccourci de l'environnement partagé sur votre espace personnel. Lorsque vous accédez au lien symbolique (/home/username/shared_env), le système vous redirige automatiquement vers le fichier ou le dossier cible, c'est-à-dire /path/to/shared_workspace. Ainsi, le contenu de shared_env est complètement identique au contenu de shared_workspace puisque si vous créer un fichier dans shared_env, celui-ci est en réalité créé dans shared_workspace. Deux utilisateur.trice.s peuvent avoir chacun.e accès à l'entièreté de l'environnement partagé et aux données qu'il contient, mais suivant les droits qui leur sont accordés, ils.elles ne peuvent écrire que dans certains dossiers.

Par exemple, dans l'image ci-dessous, user1 a le droit d'écriture dans les dossiers de projets projectA et projectB. Il.elle peut aller chercher les données dans le dossier rawdata, faire des calculs sur son espace personnel /home/user1, avec les environnements virtuels et les kernels qu'il.elle a installé, puis sauver les résultats dans projectA et projectB. Au contraire, user2 n'a les droits d'écriture que dans le dossier projectB. Cependant il.elle peut accéder aux résultats que user1 a sauvegardé dans le dossier projectA et les utiliser pour ses calculs.



Si vous souhaitez retirer le lien symbolique, il vous suffit de retourner dans le terminal et d'exécutez

```
rm shared_env
```

N.B. : Cela n'efface pas le dossier pour les autres utilisateur.trice.s, vous ne voyez simplement plus le dossier dans votre espace personnel.

Etape 2 : exécution des scripts

Indicateurs

Maintenant que l'environnement de travail est prêt, nous allons pouvoir exécuter les scripts de calculs des indicateurs/de priorisation.

Rendez-vous dans le dossier `run` présent dans `enviropace_IE`. Celui ci-contient les scripts R qui permettent de calculer les indicateurs de l'infrastructure écologique. Le fait d'utiliser un langage de programmation unique et une structure commune pour tous les scripts facilite le processus (un seul script pour un indicateur) et permet de comprendre plus facilement ce qui a été réalisé pour chacun. Le nom des scripts a été standardisé sur le modèle `ind_<pilier>_<indicateur>.r`. Pour la suite des explications, nous allons utiliser le script `ind_es_pollination.r`.

Les scripts `tools.r` permettent de prétraiter les données (si besoin) avant d'utiliser ces dernières dans les scripts d'indicateurs.

Enfin `template.r` vous permet d'écrire un nouveau script suivant le même modèle que les autres, afin de conserver une structure uniforme.

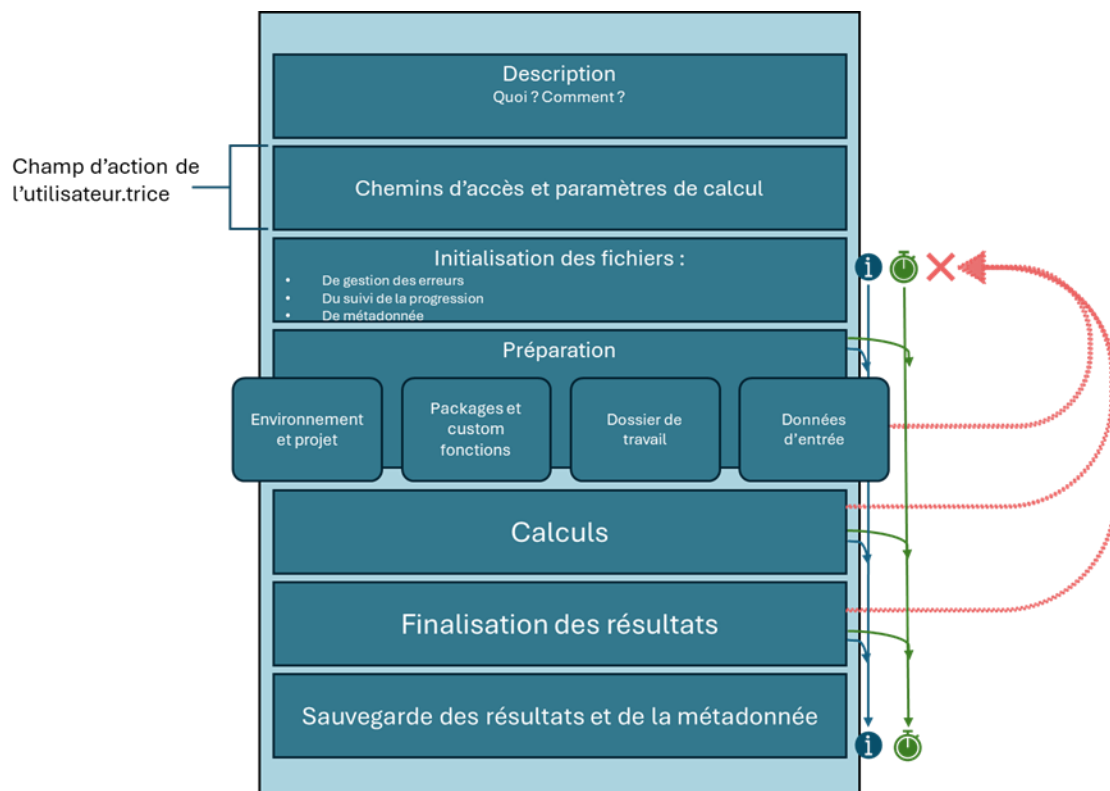
Structure générale des scripts

Il est important de **lire le script entièrement** avant de l'exécuter pour la première fois, afin de comprendre quels sont les éléments qui entrent en jeu.

Ouvrez-le et lisez-le en entier. Les scripts R du dossier run suivent tous la même structure et ont été pensés pour :

- Limiter le nombre de manipulations de la part de l'utilisateur.trice.
- Faciliter la compréhension de la méthode grâce à une nomenclature explicite et à des lignes de commentaire.
- Rendre la méthodologie choisie transparente.
- Générer automatiquement des fichiers de suivi du calcul et de documentation des résultats.
- Organiser de manière cohérente les dossiers et les résultats.

Schématiquement ils s'articulent de la manière suivante :



La première partie du script communique des informations générales sur le déroulement du processus de calcul, ainsi que le kernel nécessaire pour la bonne exécution du script.

L'utilisateur.trice interagit au niveau du second bloc, dans lequel il.elle peut modifier les données prises en entrée ainsi que les paramètres propres à la méthode suivie.

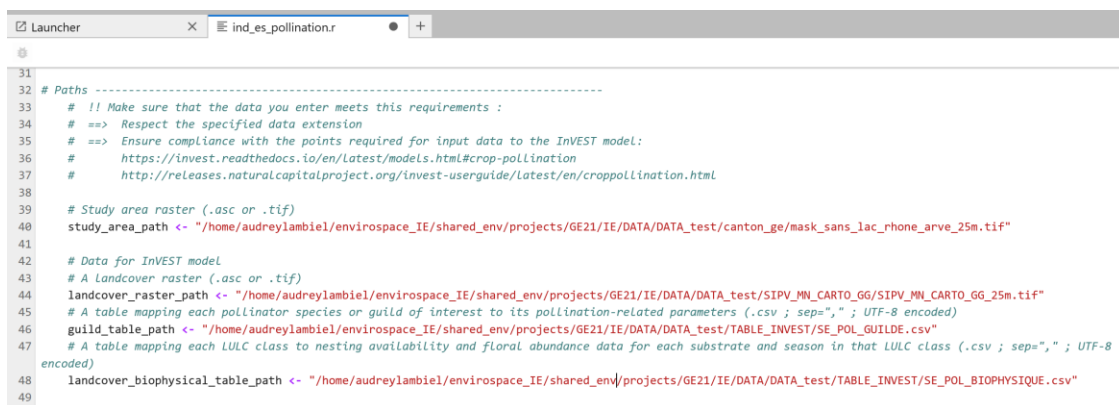
L'exécution du script génère automatiquement la métadonnée mais aussi les dossiers dans lesquels les résultats intermédiaires et finaux sont enregistrés, afin que l'architecture générale au sein du dossier projet soit conservée. Chaque étape est monitorée en termes de temps d'exécution, et en cas d'erreur, le message renvoyé est consigné dans un document pour permet le débogage.

Modifier les scripts

Les chemins d'accès aux données d'entrée nécessaires au calcul de l'indicateur doivent être listés dans la partie #Path du script. Dans notre exemple nous avons besoin :

- D'un raster masque couvrant l'air d'étude
- D'un raster de couverture du sol
- D'une table avec les préférences écologiques des guildes/des pollinisateurs considérés
- D'une table biophysique

Renseignez les chemins d'accès adéquat. Vous pouvez pour cela naviguer dans votre explorateur de fichier pour trouver la donnée, puis la sélectionner avec un clic-droit et copier le chemin d'accès en cliquant sur Copy Path. Ensuite collez le directement dans le script.



```
31
32 # Paths -----
33 # !! Make sure that the data you enter meets this requirements :
34 # ==> Respect the specified data extension
35 # ==> Ensure compliance with the points required for input data to the InVEST model:
36 # https://invest.readthedocs.io/en/latest/models.html#crop-pollination
37 # http://releases.naturalcapitalproject.org/invest-userguide/latest/en/croppollination.html
38
39 # Study area raster (.asc or .tif)
40 study_area_path <- "/home/audreylambiel/enviroSPACE_IE/shared_env/projects/GE21/IE/DATA/DATA_test/canton_ge/mask_sans_lac_rhone_arve_25m.tif"
41
42 # Data for InVEST model
43 # A Landcover raster (.asc or .tif)
44 landcover_raster_path <- "/home/audreylambiel/enviroSPACE_IE/shared_env/projects/GE21/IE/DATA/DATA_test/SIPV_MN_CARTO_GG/SIPV_MN_CARTO_GG_25m.tif"
45 # A table mapping each pollinator species or guild of interest to its pollination-related parameters (.csv ; sep="," ; UTF-8 encoded)
46 guild_table_path <- "/home/audreylambiel/enviroSPACE_IE/shared_env/projects/GE21/IE/DATA/DATA_test/TABLE_INVEST/SE_POL_GUILDE.csv"
47 # A table mapping each LULC class to nesting availability and floral abundance data for each substrate and season in that LULC class (.csv ; sep="," ; UTF-8 encoded)
48 landcover_biophysical_table_path <- "/home/audreylambiel/enviroSPACE_IE/shared_env/projects/GE21/IE/DATA/DATA_test/TABLE_INVEST/SE_POL_BIOPHYSIQUE.csv"
49
```

Dans la partie #Parameters, la section #Project vous permet de modifier certains paramètres relatifs au projet, notamment l'emplacement de l'environnement partagé, le nom du projet (ou autrement dit la version que vous produisez), l'indicateur sur lequel vous travaillez et le pilier auquel il appartient. Certains paramètres n'ont pas forcément besoin d'être modifié.

```
# Parameters -----
# Project
# Name of the main shared project folder
shared_directory <- "/home/audreylambiel/envirospace_IE/shared_env/projects/GE21/IE"
# Specify the name of an existing project or choose your new project's name
# Please note that if you enter an existing project name, previously calculated results for this indicator may be overwritten.
project_name <- "test2025"
# Name of the pillar
pillar_name <- "ES"
# Name of the indicator
indicator_name <- "POLLINATION"
# Give a short description of the indicator
description <- paste0("Capacity of ecosystems to support pollinators (InVEST model).", "\n",
  "http://releases.naturalcapitalproject.org/invest-userguide/latest/en/croppollination.html")
```

#Datas and computing parameters vous permet de renseigner les paramètres propres à la méthode appliquée (d'où l'importance de lire avant le script en entier).

```
# Datas and computing parameters
# CRS of your projet, e.g. to which your data are
mycrs <- "epsg:2056"
# Pattern used to select which of the outputs of the InVEST model should be aggregated to obtain the final result.
# See here: http://releases.naturalcapitalproject.org/invest-userguide/latest/en/croppollination.html#interpreting-results
pattern_invest_output <- "total_pollinator_abundance_"
# Function to be applied to InVEST outputs to aggregate them. Not used if only one output of the InVEST model is retained.
aggregation_function <- "sum"
```

Et enfin #Clean up options permet de gérer le contenu du dossier scratch, c'est-à-dire le dossier dans lequel les résultats intermédiaires sont enregistrés pendant le processus, et le fichier de suivi de la progression à la fin du calcul.

Paramétrez les deux options sur NO afin de pouvoir les explorer ensuite.

```
# Clean up options
# Do you want to delete the contents of the "scratch" folder at the end of the calculation? 'YES' or 'NO'
scratch_to_trash <- "NO"
# Do you want to delete the progress and error files generated while the script is running when it finishes? 'YES' or 'NO'
# n.b.: If "YES" but an error occurs, the two files will not be deleted in all cases to allow debugging.
track_to_trash <- "NO"
```

N.B. : Potentiellement d'autres options peuvent être présentes, c'est le cas par exemple pour les indicateurs du pilier structure, où une option permettant l'agrégation du raster résultat à une autre résolution est disponible. Cela dépend des scripts.

Le reste du script n'est pas censé être touché, sauf en cas de modification profonde de la méthode, ou dans le cas de bug.

Exécuter les scripts

Pour pouvoir exécuter le script, il faut dans un premier temps créer une console associée au kernel adéquat. Le kernel qui doit être utilisé pour faire tourner le script est indiqué en fin de description. Ici, le script est basé sur le modèle InVEST "Crop pollination", un modèle Python que nous allons exécuter à travers R. Nous devons donc utiliser le kernel `invest3141r` associé à l'environnement `invest3141`. Cette information se retrouve toujours en description du script.

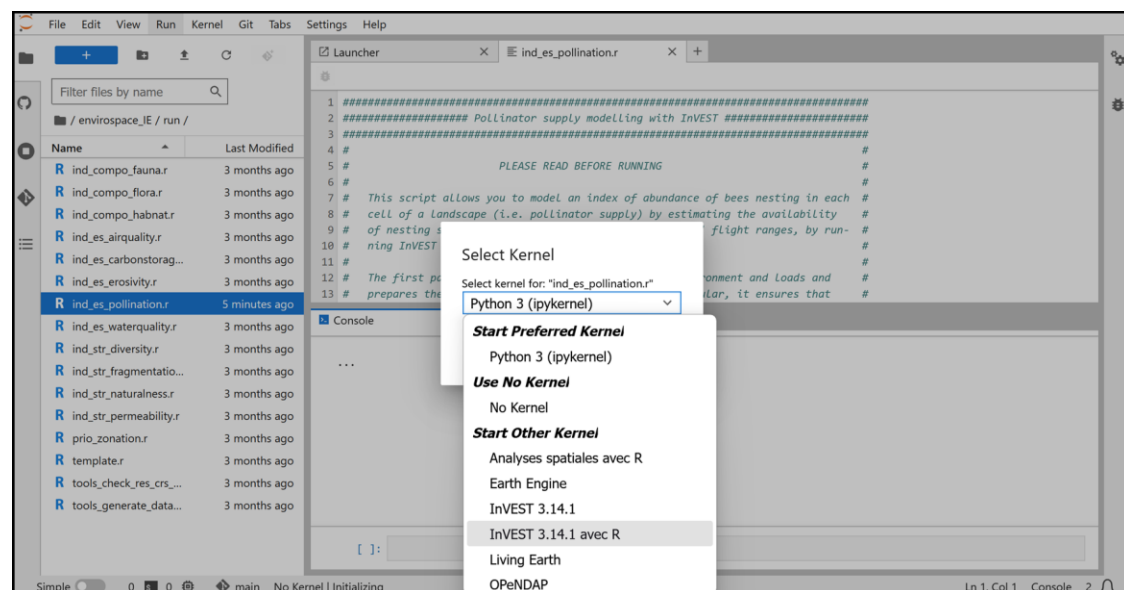
```

Launcher ind_es_pollination.r test2025_ind_es_pollination.r
14 # all the data is correctly aligned with the reference raster. #
15 # The second part of the script is used firstly to write the python script #
16 # used to launch the InVEST model, and then to read it via a system command. #
17 # The final raster is obtained by reading the output(s) of the InVEST model #
18 # and aggregating them. The final raster is clipped to the study area raster. #
19 # #
20 # The results of this script are automatically saved in the "ES" subfolder of #
21 # the project folder. Metadata file is automatically written in output folder. #
22 # For the script to run correctly, you only need to modify sections 'Paths' and #
23 # 'Parameters' below. #
24 # You don't need to make any other changes. #
25 # #
26 # Make sure you select the kernel associated with the invest3141 environment #
27 # when you run the script. #
28 # #
29 #####
30 #####
31
32 # Paths -----
33 # !! Make sure that the data you enter meets this requirements :

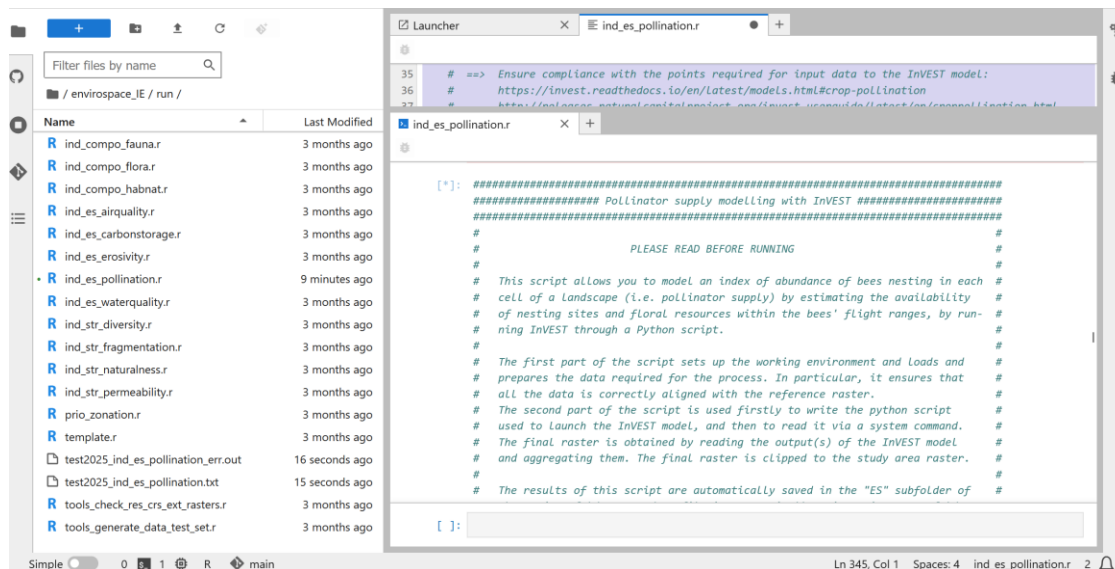
```

Faites un clic droit sur le script et sélectionnez Create Console for Editor, puis sélectionnez le kernel Invest 3.14.1 avec R.

Pour lancez le calcul de l'indicateur cliquez à nouveau sur le script et sélectionnez toutes les lignes avec crt1+A. Ensuite, exécutez le code avec shift+Enter.



Tant que [*] est présent, c'est que le code est en train de tourner. Une fois terminé, la cellule se change en [1].



Il est possible de lancer plusieurs calculs en même temps sans que ceux-ci n'interfèrent entre eux. Pour cela, suivez les mêmes étapes (lire le script ; changer les chemins d'accès/paramètres ; ouvrir une console puis exécuter). Sachez seulement que la vitesse de calcul dépend en partie des ressources du serveur et que si vous lancez plusieurs processus en même temps, les temps de calcul seront probablement rallongés. Vous pouvez également lancer des scripts, puis fermer la page du serveur/faire d'autres activités/éteindre votre ordinateur, les scripts continuent de tourner tant que vous n'éteignez pas les consoles/kernels utilisés.

Gestion des erreurs

Vous avez dû remarquer que dans votre explorateur de fichier, deux nouveaux fichiers sont apparus : celui de gestion des erreurs `test2025_ind_es_pollination_err.out`, qui devrait rester vide si tout se passe bien, et le fichier de suivis de la progression `test2025_ind_es_pollination.txt`.

En cas d'erreur dans le script ou dans son exécution, un message d'échec apparaît dans `test2025_ind_es_pollination.txt`. Le message d'erreur est quant à lui reporté dans `test2025_ind_es_pollination_err.out`.

Vous pouvez savoir que le processus est bien terminé lorsque la mention **COMPLETED** apparaît dans le fichier de suivi de la progression (ou lorsque ces deux fichiers disparaissent si vous avez paramétré `track_to_trash` sur YES).

Priorisation

Une fois que vous avez généré tous les indicateurs nécessaires à la priorisation, vous pouvez exécuter la priorisation afin de classer tous les pixels de votre zone d'étude et d'identifier votre infrastructure écologique. Pour mieux comprendre comment fonctionne [Zonation](#), nous vous invitons à lire le document associé.

Les étapes sont similaires à celles décrites dans la section Indicateurs.

Ouvrez le script `prio_zonation.r` qui se trouve dans le dossier `run` et commencez par vous assurer de bien pointer le chemin vers l'exécutable `Zonation` dans votre dossier `zonation_folder` :

```
z5_exe <- "path/to/zonation_folder/squashfs-root/usr/bin/z5"
```

Ensuite, adaptez les paramètres d'entrée en fonction des couches d'indicateurs à utiliser (par exemple ceux que vous avez générés à l'étape précédente, ou d'autres, de versions ultérieures), les poids, groupes et masques éventuels, et vérifiez tous les chemins d'accès. Une fois prêt, vous pouvez exécuter le script de la même manière que pour les scripts indicateurs.

Résultats et métadonnées

Un dossier `OUTPUTS` est créé automatiquement dans le dossier que vous avez pointé comme dossier partagé dans la section `#parameters` (dans notre exemple c'est `shared_env`), et est toujours organisé de la même manière.

```
<shared_env>/
├── OUTPUTS/
│   ├── INDICATORS/
│   │   ├── <pillar>/                                # pour chaque pilier
│   │   │   ├── <indicator>/                          # pour chaque indicateur dans le pilier
│   │   │   │   ├── <version>/                        # pour chaque version
│   │   │   │   │   ├── <result.tif>                  # résultat(s)
│   │   │   │   │   ├── METADATA.txt                 # résumé du processus et paramètres
│   │   │   │   │   └── scratch/                     # optionnel: résultats intermédiaires
│   │   └── PRIORITIZATION/
│   │       ├── <version>/                            # pour chaque version de priorisation
│   │       │   ├── zonation_output/
│   │       │   │   ├── rankmap.tif                  # raster final
│   │       │   │   ├── METADATA.txt                 # résumé du processus et paramètres
│   │       │   └── zonation_settings/
│   │       │       ├── settings.z5                  # paramètres pour exécuter Zonation
│   │       │       ├── all_files.txt                 # liste des inputs avec poids et groupes
│   │       │       └── weight_group.txt              # optionnel: fichiers de poids
```

Rendez-vous dans le dossier `OUTPUTS` et naviguez jusqu'au dossier contenant le résultat de l'indicateur `ind_es_pollination`. Un fichier texte `METADATA.txt` est présent. Cliquez sur ce dernier pour l'ouvrir.

Ce fichier a pour but de centraliser les informations importantes pour comprendre comment l'indicateur a été produit, quand, et par qui. D'un indicateur/script à l'autre, les informations stockées dans ce document varient, mais la structure globale reste la même :

- Une première partie donnant les caractéristiques de la version/projet dans lequel l'indicateur a été calculé (nom de la version/projet, date, utilisateur.trice ayant exécuté le code, indicateur)

- La liste des données ayant été prise en entrée
- Les paramètres utilisés/des informations sur la méthodologie suivie
- Des informations sur les résultats finaux (nom, emplacement de sauvegarde, résolution, CRS, etc.)
- Les temps de chaque étapes (généralement lecture et préparation des données d'entrée, calculs et finalisation) ainsi que le temps total (en fin de document)

Ressources

Lexique

Console : Environnement interactif permettant d'exécuter des commandes de code de manière séquentielle (entrer du code et obtenir des résultats immédiatement).

Conda : Gestionnaire de paquets et système de gestion d'environnements virtuels. Il est utilisé pour installer, exécuter et mettre à jour des packages et leurs dépendances, ainsi que pour créer et gérer des environnements virtuels de manière efficace.

Environnement virtuel : Espace isolé où l'on peut installer des dépendances et des packages spécifiques à un projet sans interférer avec les autres projets ou les installations globales du système. Cela permet de gérer différentes versions de packages et de bibliothèques pour différents projets.

Espace partagé : Zone de travail collaborative où plusieurs utilisateur.trice.s peuvent accéder, modifier et gérer des fichiers et des ressources en commun. Cet espace est conçu pour faciliter le travail en équipe et la collaboration sur des projets communs.

Espace personnel : Zone de travail privée où un.e utilisateur.trice peut stocker et gérer ses fichiers, notebooks et autres ressources sans que les autres utilisateur.trice.s ne puissent y accéder.

JupyterLab : Interface utilisateur interactive et flexible pour travailler avec des notebooks, du code et des données. Il est le successeur de l'interface classique des notebooks Jupyter, offrant une expérience plus intégrée et riche.

Kernel (ou noyau) : Moteur de calcul dans JupyterLab qui exécute le code en arrière-plan. Celui-ci peut être basé sur différents langages de programmation (comme Python, R, Julia, etc.). Le kernel exécute le code et renvoie le résultat.

Langage de programmation (exemple : Python, R, Julia, ...) : Ensemble de règles et de syntaxe que les développeur.euse.s utilisent pour écrire des scripts, programmes et applications. Dans JupyterLab, divers langages de programmation peuvent être disponibles.

Lien symbolique : Raccourci ou référence vers un autre fichier ou dossier sur le système de fichiers. Un lien symbolique permet d'accéder au fichier ou dossier cible à partir de plusieurs emplacements sans avoir à dupliquer les données.

Machine locale : Ordinateur personnel de l'utilisateur.trice, sur lequel JupyterLab peut être installé et exécuté. Contrairement au serveur, la machine locale est directement contrôlée par l'utilisateur.trice et les ressources y sont disponibles localement.

Notebook : Document interactif utilisé dans JupyterLab, qui combine du code, du texte, des visualisations et des médias riches dans un seul fichier.

Package (ou module) : Collections de fichiers de code qui ajoutent des fonctionnalités spécifiques à un programme ou une application. Ils peuvent contenir des bibliothèques, des outils, des scripts et des fichiers de données nécessaires pour accomplir certaines tâches dans un langage de programmation.

Serveur : Machine (physique ou virtuelle) qui héberge l'instance de JupyterLab et permet aux utilisateur.trice.s d'y accéder via un navigateur web. Le serveur gère les requêtes des utilisateur.trice.s, exécute les kernels et permet la gestion des fichiers et des ressources.

Terminal : Interface de ligne de commande qui permet d'exécuter des commandes du système d'exploitation directement.

Quelques commandes utiles

Gestion des environnements avec Conda

Créer un environnement

```
conda create -n myenv
```

Créer un environnement à partir d'un fichier .yaml

```
conda env create -f myenv.yaml
```

Activer un environnement

```
conda activate myenv
```

Désactiver un environnement

```
conda deactivate
```

Lister les environnements disponibles

```
conda info --envs
```

Supprimer un environnement

```
conda env remove --name myenv
```

Exporter un environnement en un fichier .yaml

```
conda env export > myenv.yaml
```

Cloner un environnement

```
conda create --name myclone --clone myenv
```

Gestion des packages

Lister les packages

```
conda list
```

Installer un package

```
conda install package_name
```

Installer avec un canal spécifique

```
conda install -c conda-forge package_name
```

Mettre à jour un package

```
conda update package_name
```

Supprimer un package

```
conda remove package_name
```

Kernels Jupyter

Lister les kernels

```
jupyter kernelspec list
```

Créer un kernel R

```
IRkernel::installspec(name = 'nom_kernel', displayname = 'Nom affiché')
```

Créer un kernel Python

```
python -m ipykernel install --user --name nom_kernel -- display-name "Nom affiché"
```

Supprimer un kernel

```
jupyter kernelspec uninstall nom_kernel
```

Gestion des dossiers via le terminal

Supprimer un dossier non-vidé

```
rm -r nom_dossier
```

Zippé un dossier

```
shutil.make_archive("chemin/nom_zip", "zip", "chemin/dossier")
```

Dézipper un dossier

```
shutil.unpack_archive("chemin/zip", "chemin/cible", "zip")
```

Références et liens utiles

Article de référence associé

Lambiel, Audrey, Gregory Giuliani, Nathan Külling, and Anthony Lehmann. In prep. "A Digital Twin approach for the identification and update of Ecological Infrastructure".

Répertoire GitHub du projet

https://github.com/ALambiel/enviropace_IE

Articles scientifiques liés

- Honeck, Erica, Louise Gallagher, Bertrand von Arx, et al. 2021. "Integrating Ecosystem Services into Policymaking – A Case Study on the Use of Boundary Organizations." *Ecosystem Services* 49 (June): 101286. <https://doi.org/10.1016/j.ecoser.2021.101286>.
- Honeck, Erica, Atte Moilanen, Benjamin Guinaudeau, et al. 2020. "Implementing Green Infrastructure for the Spatial Planning of Peri-Urban Areas in Geneva, Switzerland." *Sustainability* 12 (4): 4. <https://doi.org/10.3390/su12041387>.
- Honeck, Erica, Arthur Sanguet, Martin A. Schlaepfer, Nicolas Wyler, and Anthony Lehmann. 2020. "Methods for Identifying Green Infrastructure." *SN Applied Sciences* 2 (11): 1916. <https://doi.org/10.1007/s42452-020-03575-4>.
- Moilanen, Atte, Pauli Lehtinen, Ilmari Kohonen, Joel Jalkanen, Elina A. Virtanen, and Heini Kujala. 2022. "Novel Methods for Spatial Prioritization with Applications in Conservation, Land Use Planning and Ecological Impact Avoidance." *Methods in Ecology and Evolution* 13 (5): 1062–72. <https://doi.org/10.1111/2041-210X.13819>.
- Moilanen, Atte, Kerrie A. Wilson, and Hugh P. Possingham, eds. 2009. *Spatial Conservation Prioritization: Quantitative Methods and Computational Tools*. Oxford University Press. <https://doi.org/10.1093/oso/9780199547760.001.0001>.
- Sanguet, Arthur, Nicolas Wyler, Benjamin Guinaudeau, Noé Waller, Loreto Urbina, Laurent Huber, Claude Fischer, and Anthony Lehmann. 2023. "Mapping Ecological Infrastructure in a Cross-Border Regional Context." *Land* 12 (11): 2010. <https://doi.org/10.3390/land12112010>.
- Urbina, Loreto, Anthony Lehmann, Laurent Huber, and Claude Fischer. 2024. "Combining multi-species connectivity modelling with expert knowledge to inform the green infrastructure design." *Journal for Nature Conservation*, 81: 126654. <https://doi.org/https://doi.org/10.1016/j.jnc.2024.126654>

Documentation technique

JupyterLab Documentation - Contient des informations détaillées sur l'installation, l'utilisation et la personnalisation de JupyterLab.

Conda User Guide - Tasks - Décrit les différentes tâches que vous pouvez effectuer avec Conda, telles que la gestion des environnements et des paquets.

Conda Cheatsheet - Fournit un résumé synthétique des commandes de base pour la gestion des environnements, l'installation de paquets, et l'import/export des environnements avec Conda.

Installation de kernels Jupyter pour divers langages - Guide pour installer des kernels Jupyter pour Java, Scala, Go, Rust et R.

Gestion des fichiers et répertoires avec Bash - Couvre les commandes Bash essentielles pour gérer les fichiers et répertoires.