Traitement d’images satellites avec Python

Première édition

Philippe Apparicio

Yacine Bouroubi

Samuel Foucher

Mickaël Germain

2024-09-29

Table des matières

# Préface

**Résumé :** Ce livre vise à décrire une panoplie de méthodes de traitement d’images satellites avec le langage Python. Celles et ceux souhaitant migrer progressivement d’un autre logiciel d’imagerie et de télédétection vers Python trouveront dans cet ouvrage les éléments pour une transition en douceur. La philosophie de ce livre est de donner toutes les clefs de compréhension et de mise en œuvre des méthodes abordées dans Python. La présentation des méthodes est basée sur une approche compréhensive et intuitive plutôt que mathématique, sans pour autant négliger la rigueur mathématique ou statistique.

**Remerciements :** Ce manuel a été réalisé avec le soutien de la fabriqueREL. Fondée en 2019, la fabriqueREL est portée par divers établissements d’enseignement supérieur du Québec et agit en collaboration avec les services de soutien pédagogique et les bibliothèques. Son but est de faire des ressources éducatives libres (REL) le matériel privilégié en enseignement supérieur au Québec.

**Maquette de la page couverture et identité graphique du livre :** Andrés Henao Florez.

**Mise en page :** Philippe Apparicio et Marie-Hélène Gadbois Del Carpio.

**Révision linguistique :** Denise Latreille.

© Philippe Apparicio, Yacine Bouroubi, Samuel Foucher et Mickël Foucher.

**Pour citer cet ouvrage :** Apparicio P., Bouroubi, Y., Foucher, S. et M. Germain (2024). *Traitement d’images satellites avec Python*. Université de Sherbrooke, Département de géomatique appliquée. fabriqueREL. Licence CC BY-SA.



## Un manuel sous la forme d’une ressource éducative libre

**Pourquoi un manuel sous licence libre?**

Les logiciels libres sont aujourd’hui très répandus. Comparativement aux logiciels propriétaires, l’accès au code source permet à quiconque de l’utiliser, de le modifier, de le dupliquer et de le partager. Le logiciel Python, dans lequel sont mises en œuvre les méthodes de traitement d’images satellites décrites dans ce livre, est d’ailleurs à la fois un langage de programmation et un logiciel libre (sous la licence publique générale [GNU GPL2](https://fr.wikipedia.org/wiki/Licence_publique_g%C3%A9n%C3%A9rale_GNU)). Par analogie aux logiciels libres, il existe aussi des **ressources éducatives libres (REL)** « dont la licence accorde les permissions désignées par les 5R (**Retenir — Réutiliser — Réviser — Remixer — Redistribuer**) et donc permet nécessairement la modification » ([***fabriqueREL***](https://fabriquerel.org/rel/)). La licence de ce livre, CC BY-SA ([figure 1](#fig-Licence)), permet donc de :

* **Retenir**, c’est-à-dire télécharger et imprimer gratuitement le livre. Notez qu’il aurait été plutôt surprenant d’écrire un livre payant sur un logiciel libre et donc gratuit. Aussi, nous aurions été très embarrassés que des personnes étudiantes avec des ressources financières limitées doivent payer pour avoir accès au livre, sans pour autant savoir préalablement si le contenu est réellement adapté à leurs besoins.
* **Réutiliser**, c’est-à-dire utiliser la totalité ou une section du livre sans limitation et sans compensation financière. Cela permet ainsi à d’autres personnes enseignantes de l’utiliser dans le cadre d’activités pédagogiques.
* **Réviser**, c’est-à-dire modifier, adapter et traduire le contenu en fonction d’un besoin pédagogique précis puisqu’aucun manuel n’est parfait, tant s’en faut! Le livre a d’ailleurs été écrit intégralement dans R avec [Quatro](https://quarto.org/). Quiconque peut ainsi télécharger gratuitement le code source du livre sur [github](https://github.com/SerieBoldR/MethodesAnalyseSpatiale) et le modifier à sa guise (voir l’encadré intitulé *Suggestions d’adaptation du manuel*).
* **Remixer**, c’est-à-dire « combiner la ressource avec d’autres ressources dont la licence le permet aussi pour créer une nouvelle ressource intégrée » ([***fabriqueREL***](https://fabriquerel.org/rel/)).
* **Redistribuer**, c’est-à-dire distribuer, en totalité ou en partie le manuel ou une version révisée sur d’autres canaux que le site Web du livre (par exemple, sur le site Moodle de votre université ou en faire une version imprimée).

La licence de ce livre, CC BY-SA ([figure 1](#fig-Licence)), oblige donc à :

* Attribuer la paternité de l’auteur dans vos versions dérivées, ainsi qu’une mention concernant les grandes modifications apportées, en utilisant la formulation suivante :

Apparicio Philippe, Yacine Bouroubi, Samuel Foucher et Mickaël Foucher (2024). *Traitement d’images satellites :* . Université de Sherbrooke, Département de géomatique appliquée. fabriqueREL. Licence CC BY-SA.

* Utiliser la même licence ou une licence similaire à toutes versions dérivées.

|  |
| --- |
| Figure 1: Licence Creative Commons du livre |

**Suggestions d’adaptation du manuel**

Pour chaque méthode d’analyse spatiale abordée dans le livre, une description détaillée et une mise en œuvre dans Python sont disponibles. Par conséquent, plusieurs adaptations du manuel sont possibles :

* Conserver uniquement les chapitres sur les méthodes ciblées dans votre cours.
* En faire une version imprimée et la distribuer aux personnes étudiantes.
* Modifier la description d’une ou de plusieurs méthodes en effectuant les mises à jour directement dans les chapitres.
* Insérer ses propres jeux de données dans les sections intitulées *Mise en œuvre dans Python*.
* Modifier les tableaux et figures.
* Ajouter une série d’exercices.
* Modifier les quiz de révision.
* Rédiger un nouveau chapitre.
* Modifier des syntaxes en Python. Plusieurs *librairies* Python peuvent être utilisées pour mettre en œuvre telle ou telle méthode. Ces derniers évoluent aussi très vite et de nouvelles *librairies* sont proposées fréquemment! Par conséquent, il peut être judicieux de modifier une syntaxe Python du livre en fonction de ses habitudes de programmation en Python (utilisation d’autres *librairies* que ceux utilisés dans le manuel par exemple) ou de bien mettre à jour une syntaxe à la suite de la parution d’une nouvelle *librairie* plus performante ou intéressante.
* Toute autre adaptation qui permet de répondre au mieux à un besoin pédagogique.

## Comment lire ce manuel?

Le livre comprend plusieurs types de blocs de texte qui en facilitent la lecture.

**Bloc *packages***

Habituellement localisé au début d’un chapitre, il comprend la liste des *packages* Python utilisés pour un chapitre.

**Bloc objectif**

Il comprend une description des objectifs d’un chapitre ou d’une section.

**Bloc notes**

Il comprend une information secondaire sur une notion, une idée abordée dans une section.

**Bloc pour aller plus loin**

Il comprend des références ou des extensions d’une méthode abordée dans une section.

**Bloc astuce**

Il décrit un élément qui vous facilitera la vie : une propriété statistique, un *package*, une fonction, une syntaxe Python.

**Bloc attention**

Il comprend une notion ou un élément important à bien maîtriser.

**Bloc exercice**

Il comprend un court exercice de révision à la fin de chaque chapitre.

## Comment utiliser les données du livre pour reproduire les exemples?

Ce livre comprend des exemples détaillés et appliqués dans R pour chacune des méthodes abordées. Ces exemples se basent sur des jeux de données structurés et mis à disposition avec le livre. Ils sont disponibles sur le *repo github* dans le sous-dossier data, à l’adresse <https://github.com/SerieBoldR/TraitementImagesVol1/tree/main/data>.

Une autre option est de télécharger le *repo* complet du livre directement sur *github* (<https://github.com/SerieBoldR/TraitementImagesVol1>) en cliquant sur le bouton Code, puis le bouton Download ZIP ([figure 2.1](#fig-downloaffromgit)). Les données se trouvent alors dans le sous-dossier nommé data.

|  |
| --- |
| Figure 2: Téléchargement de l’intégralité du livre |

## Structure du livre

Le livre est organisé autour de quatre grandes parties.

**Partie 1. Importation et manipulation de données spatiales.** Dans cette première partie, nous voyons comment importer, manipuler, cartographier et exporter des données spatiales dans R, principalement avec les *packages* sf pour les données vectorielles, terra pour les données matricielles (images) et tmap pour la cartographie ([chapitre 2](#sec-chap01)). Maîtriser les notions abordées dans ce chapitre constitue une étape préalable et indispensable à tout projet d’analyse spatiale. D’une part, avant d’analyser des données spatiales, il convient de les structurer (importation et manipulation) et de les explorer (cartographie). D’autre part, une fois la ou les méthodes d’analyse spatiale mises en œuvre, il convient de cartographier les résultats finaux et de les exporter au besoin dans un format de données géographiques (shapefile (shp), GeoPackage (GPKG), GeoJSON (geojson), sqlite (sqlite), GeoTiff, etc.).

**Partie 2. Transformations des données spatiales**. Cette troisième partie comprend deux chapitres : les transformations spectrales ([chapitre 4](#sec-chap03)) et les transformations spatiales ([chapitre 5](#sec-chap04)).

**Partie 3. Classifications d’images.** Cette troisième partie comprend deux chapitres : les classifications supervisées ([chapitre 6](#sec-chap05)) et non supervisées ([chapitre 7](#sec-chap06)).

**Partie 4. Données massives**. Cette quatrième et dernière partie comprend un seul chapitre qui est dédié aux plateformes de mégadonnes [chapitre 8](#sec-chap07), notammment Google Earth Engine.

## Remerciements

De nombreuses personnes ont contribué à l’élaboration de ce manuel.

Ce projet a bénéficié du soutien pédagogique et financier de la [***fabriqueREL***](https://fabriquerel.org/) (ressources éducatives libres). Les différentes rencontres avec le comité de suivi nous ont permis de comprendre l’univers des ressources éducatives libres (REL) et notamment leurs [fameux 5R](https://fabriquerel.org/rel/) (Retenir — Réutiliser — Réviser — Remixer — Redistribuer), de mieux définir le besoin pédagogique visé par ce manuel, d’identifier des ressources pédagogiques et des outils pertinents pour son élaboration. Ainsi, nous remercions chaleureusement les membres de la *fabriqueREL* pour leur soutien inconditionnel :

* Myriam Beaudet, bibliothécaire à l’Université de Sherbrooke.
* Marianne Dubé, coordonnatrice de la fabriqueREL, Université de Sherbrooke.
* Serge Piché, conseiller pédagogique, Université de Sherbrooke.
* Claude Potvin, conseiller en formation, Service de soutien à l’enseignement, Université Laval.

Nous remercions chaleureusement les personnes étudiantes des cours **à modifier plus tard** du [Baccalauréat en géomatique appliquée à l’environnement](https://www.usherbrooke.ca/admission/programme/271/baccalaureat-en-geomatiqueappliquee-a-lenvironnement/) et du [Microprogramme de 1er cycle en géomatique appliquée](https://www.usherbrooke.ca/admission/programme/429/microprogramme-de-1er-cycleen-geomatique-appliquee/) du [Département de géomatique appliquée](https://www.usherbrooke.ca/geomatique/) de l’[Université de Sherbrooke](https://www.usherbrooke.ca/) de la session d’été 2023 : à modifier plus tard.

Nous remercions aussi les membres du comité de révision pour leurs commentaires et suggestions très constructifs. Ce comité est composé de quatre personnes étudiantes du [Département de géomatique appliquée](https://www.usherbrooke.ca/geomatique/) de l’[Université de Sherbrooke](https://www.usherbrooke.ca/) :

* À compléter plus tard.
* À compléter plus tard.

Finalement, nous remercions Denise Latreille, réviseure linguistique et chargée de cours à l’Université Sherbrooke, pour la révision du manuel.

## Introduction aux images de télédétection

L’imagerie numérique a pris une place importante dans notre vie de tous les jours depuis une quinzaine d’année. Ces images sont prises généralement au niveau du sol (imagerie proximale) avec seulement trois couleurs dans le domaine de la vision humaine (rouge, vert et bleu). Dans la suite du manuel, on parlera d’images du domaine de la vision par ordinateur ou images en vision pour faire plus court.

Les images de télédétection ont des particularités et des propriétés qui les différencient des images de tous les jours. On peut souligner au moins cinq caractéristiques principales: 1. Les images sont géoréférencées : Cela veut dire que pour chaque pixel nous pouvons y associer une position géographique ou cartographique. 2. Le point de vue est très différent : Ces images sont prises avec une vue d’en haut (Nadir) ou oblique avec une distance qui peut être très grande (On parle d’images distales). 3. Elles possèdent plus que 3 bandes : Contrairement aux images en vision, les images de télédétection possèdent bien souvent plus que 3 bandes. Il n’est pas rare de trouver 4 bandes (Pléiade), 13 bandes (Sentinel-2, Landsat) et même 200 bandes pour des capteurs hyperspectraux. 4. Elles peuvent être calibrées : Les valeurs numérique de l’image peuvent être converties en quantités physiques (luminance, réflectance, section efficiace, etc.) via une fonction de calibration. 5. Elles sont de grande taille : Il n’est pas rare de manipuler des images qui font plusieurs dizaines de milliers de pixels en dimension.

### Ressources en ligne

### Listes des *librairies* utilisés

Dans ce livre, nous utilisons de nombreux *packages* Python que vous pouvez installer avec le code ci-dessous.

import sys  
import subprocess  
import pkg\_resources  
  
# Liste des packages  
list\_packages = ["pandas", "scikit-image", "matplotlib",   
 "geopandas", "rasterio", "folium"]  
  
# Packages non installés dans la liste  
installed\_packages = {pkg.key for pkg in pkg\_resources.working\_set}  
packages\_not\_installed = [pkg for pkg in list\_packages if pkg.lower() not in installed\_packages]  
  
# Installation des packages manquants  
if packages\_not\_installed:  
 print("Installing missing packages...")  
 for package in packages\_not\_installed:  
 subprocess.check\_call([sys.executable, "-m", "pip", "install", package])  
 print("All packages installed successfully.")  
else:  
 print("All required packages are already installed.")

### Python 101

|  |
| --- |
| import numpy as np import matplotlib.pyplot as plt  r = np.arange(0, 2, 0.01) theta = 2 \* np.pi \* r fig, ax = plt.subplots(  subplot\_kw = {'projection': 'polar'}  ) ax.plot(theta, r) ax.set\_rticks([0.5, 1, 1.5, 2]) ax.grid(True) plt.show()  Figure 3 |

# À propos des auteurs

[**Philippe Apparicio**](https://www.usherbrooke.ca/recherche/fr/specialistes/details/philippe.apparicio) est professeur titulaire au [Département de géomatique appliquée](https://www.usherbrooke.ca/geomatique/) de l’[Université de Sherbrooke](https://www.usherbrooke.ca/). Il y enseigne aux [programmes de 1er et 2e cycles de géomatique](https://www.usherbrooke.ca/geomatique/etudes/programmes) les cours *Transport et mobilité durable*, *Modélisation et analyse spatiale* et *Géomatique appliquée à la gestion urbaine*. Durant les dernières années, il a offert plusieurs formations aux Écoles d’été du Centre interuniversitaire québécois de statistiques sociales ([CIQSS](https://www.ciqss.org/)). Géographe de formation, ses intérêts de recherche incluent la justice et l’équité environnementale, la mobilité durable, les pollutions atmosphérique et sonore, et le vélo en ville. Il a publié une centaine d’articles scientifiques dans différents domaines des études urbaines et de la géographie mobilisant la géomatique et l’analyse spatiale.

[**Yacine Bouroubi**](https://www.usherbrooke.ca/geomatique/departement/personnel/personnel-enseignant/yacine-bouroubi) est professeur titulaire au [Département de géomatique appliquée](https://www.usherbrooke.ca/geomatique/) de l’[Université de Sherbrooke](https://www.usherbrooke.ca/). Il y enseigne aux [programmes de 1er et 2e cycles de géomatique](https://www.usherbrooke.ca/geomatique/etudes/programmes) les cours *Transport et mobilité durable*, *Modélisation et analyse spatiale* et *Géomatique appliquée à la gestion urbaine*. Durant les dernières années, il a offert plusieurs formations aux Écoles d’été du Centre interuniversitaire québécois de statistiques sociales ([CIQSS](https://www.ciqss.org/)). Géographe de formation, ses intérêts de recherche incluent la justice et l’équité environnementale, la mobilité durable, les pollutions atmosphérique et sonore, et le vélo en ville. Il a publié une centaine d’articles scientifiques dans différents domaines des études urbaines et de la géographie mobilisant la géomatique et l’analyse spatiale.

[**Samuel Foucher**](https://www.usherbrooke.ca/recherche/fr/specialistes/details/samuel.foucher) est professeur titulaire au [Département de géomatique appliquée](https://www.usherbrooke.ca/geomatique/) de l’[Université de Sherbrooke](https://www.usherbrooke.ca/). Il y enseigne aux [programmes de 1er et 2e cycles de géomatique](https://www.usherbrooke.ca/geomatique/etudes/programmes) les cours *Transport et mobilité durable*, *Modélisation et analyse spatiale* et *Géomatique appliquée à la gestion urbaine*. Durant les dernières années, il a offert plusieurs formations aux Écoles d’été du Centre interuniversitaire québécois de statistiques sociales ([CIQSS](https://www.ciqss.org/)). Géographe de formation, ses intérêts de recherche incluent la justice et l’équité environnementale, la mobilité durable, les pollutions atmosphérique et sonore, et le vélo en ville. Il a publié une centaine d’articles scientifiques dans différents domaines des études urbaines et de la géographie mobilisant la géomatique et l’analyse spatiale.

[**Mickaël Germain**](https://www.usherbrooke.ca/geomatique/departement/personnel/personnel-enseignant/mickael-germain) est professeur titulaire au [Département de géomatique appliquée](https://www.usherbrooke.ca/geomatique/) de l’[Université de Sherbrooke](https://www.usherbrooke.ca/). Il y enseigne aux [programmes de 1er et 2e cycles de géomatique](https://www.usherbrooke.ca/geomatique/etudes/programmes) les cours *Transport et mobilité durable*, *Modélisation et analyse spatiale* et *Géomatique appliquée à la gestion urbaine*. Durant les dernières années, il a offert plusieurs formations aux Écoles d’été du Centre interuniversitaire québécois de statistiques sociales ([CIQSS](https://www.ciqss.org/)). Géographe de formation, ses intérêts de recherche incluent la justice et l’équité environnementale, la mobilité durable, les pollutions atmosphérique et sonore, et le vélo en ville. Il a publié une centaine d’articles scientifiques dans différents domaines des études urbaines et de la géographie mobilisant la géomatique et l’analyse spatiale.

# 1. Introduction au langage Python

Dans ce chapitre, nous allons présenter quelques éléments essentiels du langage Python qui nous seront utiles dans ce manuel. Python est un langage très riche et peut aboutir à des projets logiciels très sophistiqués. Il est important de comprendre que la programmation Python n’est pas une fin en soit ici. Python est pour nous principalement un outil de ‘scriptage’ et de manipulation de la donnée.

Python, créé par [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) en 1991, est un langage de programmation polyvalent et facile à apprendre, souvent comparé à un couteau suisse numérique pour sa simplicité et sa polyvalence. Comme un outil multifonction, Python peut être utilisé pour une variété de tâches, du développement web à l’analyse de données, en passant par l’intelligence artificielle.

## 1.1 Les distributions

Il existe plusieurs [distributions](https://wiki.python.org/moin/PythonDistributions) du langage Python, ces distributions sont comme différentes saveurs de votre glace préférée - chacune a ses propres caractéristiques uniques, mais elles sont toutes fondamentalement Python. Voici un aperçu des principales distributions :

* [CPython](https://www.python.org/downloads/) : C’est la distribution “vanille” officielle, comme la recette originale de Python. C’est le choix idéal pour la compatibilité et la conformité aux standards.
* [Anaconda](https://www.anaconda.com/download) : Pensez-y comme à un sundae tout garni. Il vient avec de nombreuses bibliothèques scientifiques préinstallées, idéal pour l’analyse de données et le machine learning.
* [Miniconda](https://docs.anaconda.com/miniconda/miniconda-install/) : est une distribution légère de Python qui vous permet d’ajouter les librairies au besoin.
* PyPy : C’est comme une version turbo de Python, optimisée pour la vitesse.

Jython et IronPython : Ces versions sont comme des traducteurs, permettant à Python de “parler” Java (Jython) ou .NET (IronPython). Chaque distribution a ses forces, que ce soit la simplicité, la vitesse ou des fonctionnalités spécifiques. Le choix dépend de vos besoins, comme choisir entre une glace simple ou un banana split élaboré.

## 1.2 Les styles de programmation en Python

Il existe plusieurs approches pour programmer en Python. La plus directe est en version interactive en tapant python et de rentrer des commandes ligne par ligne.

### 1.2.1 Les outils de programmation

Un code python prend la forme d’un simple fichier texte avec l’extension .py et peut être modifié avec un simple éditeur de texte. Cependnant, il n’y aura pas de rétroactions immédiates de l’interpréteur Python ce qui rend la correction d’erreurs (débogage) beaucoup plus laborieux.

Un IDE (*Integrated Developement Environnement*) est comme une boîte à outils complète pour les programmeurs, vous trouverez :

* Un éditeur de texte amélioré pour écrire votre code, avec des fonctionnalités comme la coloration syntaxique qui rend le code plus lisible.
* Un compilateur qui transforme votre code en instructions que l’ordinateur peut comprendre.
* Un débogueur pour trouver et corriger les erreurs, tel un détective numérique.
* Des outils d’automatisation qui effectuent des tâches répétitives, comme un assistant virtuel pour le codage.
* L’accès à la documentation des différentes librairies.

Ces outils intégrés permettent aux développeurs de travailler plus efficacement, en passant moins de temps à jongler entre différentes applications et plus de temps à produire du code.

Voici quelques options populaires :

* [PyCharm](https://www.jetbrains.com/pycharm/) : C’est un des outils les plus utilisés dans l’industrie. Il offre une multitude de fonctionnalités comme l’autocomplétion intelligente et le débogage intégré, idéal pour les grands projets. Cepednant, cet outil peut être assez gourmand en mémoire et en CPU.
* [Visual Studio Code](https://code.visualstudio.com/) : Gratuit, léger mais puissant, il est personnalisable avec des extensions pour Python.
* [Spyder](https://www.spyder-ide.org/) : Logiciel libre et gratuit, orienté vers les applications scientifiques.
* [Jupyter Notebooks](https://jupyter.org/) : Imaginez un cahier interactif pour le code. Idéal pour l’analyse de données et l’apprentissage, il permet de mélanger code, texte et visualisations. Des services gratuits dans le **cloud** sont disponibles comme Google Colab et Kaggle. Ces environnements sont néanmoins moins appropriées pour des grands projets et le débogage.
* Sublime Text : C’est comme un stylo élégant et rapide. Léger et rapide, il est apprécié pour sa simplicité et sa vitesse. Le choix dépend de vos besoins, que vous soyez débutant ou développeur chevronné. L’important est de trouver l’éditeur qui vous convient le mieux pour coder confortablemen

## 1.3 Bonnes pratiques

Python est un langage très dynamique, qui évolue constamment. Il est fortement conseillé d’utiliser des environnements virtuels pour gérer vos différentes librairies. Voici quelques bonnes pratiques à suivre :

1. **N’installez par la dernière version de Python** : installez toujours une version ou deux qui précède [la dernière version](https://www.python.org/downloads/). Les versions trop récentes peuvent être instables. La version de python désirée peut être spécifiée au moment de la création d’un environnement virtuel (voir plus bas).
2. **Utilisez des environnements virtuels** : Pensez-y comme à des compartiments séparées pour chaque projet. Cela évite les conflits entre les différentes versions de bibliothèques et garde votre système propre.
3. **Vérifiez l’installation** : Après l’installation, ouvrez un terminal et tapez python --version pour vous assurer que tout fonctionne correctement.

### 1.3.1 Création d’un environnement virtuel

Il y a deux façons d’installer environnement virtuel selon votre distribution de Python:

1. **Option 1** : vous utilisez [Anaconda](https://www.anaconda.com/download) ou [Miniconda](https://docs.anaconda.com/miniconda/miniconda-install/), dans ce cas la commande conda est utilisée pour créer un environnement test avec Python 3.10:

conda env -n test python=3.10  
conda activate test

1. **Option 2** : vous utilisez [CPython](https://www.python.org/downloads/)

conda env -n test python=3.10  
conda activate test

Note: si vous utiliser des notebook jupyter, vous avez déjà un environnement virtuel qui contient le serveur jupyter.

## 1.4 Les structures de base

Il y a essentiellement deux structures de données que Python manipule : les listes et les dictionnaires.

## 1.5 Les listes

Les listes sont comme des boites extensibles où vous pouvez ranger différents types d’objets :

* Représentées par des crochets : [1, 2, 3, "python"].
* Ordonnées et modifiables (mutables), vous pouvez récupérer une valeur par sa position avec [].
* Permettent les doublons (deux fois la même valeur).
* Idéales pour stocker des collections d’éléments que vous voulez modifier

### 1.5.1 Les tuples

Les tuples sont similaires aux listes, mais les boîtes sont scellées :

* Représentés par des parenthèses : (1, 2, 3, "python").
* Ordonnés mais non modifiables (immutables).
* Permettent les doublons.
* Souvent utilisé pour stocker des données qui ne doivent pas changer (comme des paramètres).

### 1.5.2 Les ensembles (Sets)

Les ensembles sont comme des boites magiques qui ne gardent qu’un exemplaire de chaque objet :

* Représentés par des accolades : {1, 2, 3}.
* Non ordonnés et modifiables.
* N’autorisent pas les doublons.
* Utiles pour éliminer les doublons et effectuer des opérations mathématiques sur des ensembles.

## 1.6 Dictionnaires

Les dictionnaires sont comme des boites avec des étiquettes sur chcune d’elle :

* Représentés par des accolades avec des paires clé-valeur : {"nom": "Python", "année": 1991}.
* Non ordonnés et modifiables.
* Les clés doivent être uniques, mais les valeurs peuvent être dupliquées
* Utiles pour stocker des données associatives ou pour créer des tables de recherche rapide

## 1.7 Progrmmation objet

La programmation orientée objet (POO) en Python est comme construire avec des blocs LEGO. Chaque objet est un bloc LEGO avec ses propres caractéristiques (attributs) et capacités (méthodes). Les classes sont les plans pour créer ces blocs. Par exemple, une classe “Voiture” pourrait avoir des attributs comme “couleur” et “vitesse”, et des méthodes comme “démarrer” et “accélérer”.

Python rend la POO accessible avec des fonctionnalités conviviales :

1. **Encapsulation** : Comme emballer un cadeau, elle cache les détails internes d’un objet.
2. **Héritage** : Permet de créer de nouvelles classes basées sur des classes existantes, comme un enfant héritant des traits de ses parents.
3. **Polymorphisme** : Permet à différents objets de répondre au même message de manière unique, comme si différents animaux répondaient différemment à “fais du bruit”.

Ces caractéristiques font de Python un excellent choix pour apprendre et appliquer les concepts de la POO, rendant le code plus organisé et réutilisable

**Liste des *packages* utilisés dans ce chapitre**

* Pour importer et manipuler des fichiers géographiques :
  + sf pour importer et manipuler des données vectorielles.
  + terra pour importer et manipuler des données matricielles.
* Pour construire des cartes et des graphiques :
  + tmap est certainement le meilleur *package* pour la cartographie.
  + ggplot2 pour construire des graphiques.

## 1.8 Quiz de révision du chapitre

## 1.9 Cahier de révision (notebook)

**Notebook 1.** À compléter

# ...  
# à compléter

# 2. Importation et manipulation de données spatiales

Dans le chapitre, nous abordons quelques formats d’images ainsi que leur lecture.

**Liste des *librairies* utilisées dans ce chapitre**

* Pour importer et manipuler des fichiers géographiques :
  + sf pour importer et manipuler des données vectorielles.
  + Rasterio pour importer et manipuler des données matricielles.
* Pour construire des cartes et des graphiques :
  + tmap est certainement le meilleur *package* pour la cartographie.
  + Matplotlib pour construire des graphiques.

## 2.1 Bases de Quarto…

Voici comment faire une liste :

* **texte en gras**
* *texte en italique*
  + ***Gras et italique***
* R2 et NO2
* Petites majuscules
* Pour un lien Web, [Département de géomatique appliquée](https://www.usherbrooke.ca/geomatique/).

Voici comment intégrer des équations LaTeX. La formule de la distance euclidienne ([équation 2.2](#eq-DistEuc)). Pour écrire directement une équation dans le texte, il suffit de .

* Intégrer une figure (image) et l’appeler dans le texte ([figure 2.1](#fig-downloaffromgit)). À la [figure 2.1](#fig-downloaffromgit), … Notez que la numérotation des figures, des tableaux, des équations est automatique.

|  |
| --- |
| Figure 2.1: Téléchargement de l’intégralité du livre |

* Les références sont au format BibTeX. Par exemple, vous pouvez les télécharger de Google Scholar et les intégrer à la fin du fichier references.bib. Voici comment intégrer des références (Mather et Koch 2022; Richards, Richards et al. 2022). Selon Ferdinand Boon et Guy Rochon (1992).

Le code R ci-dessous permet de faire un tableau dans un chunk! Pour appeler le tableau (**?@tbl-TabMatricesSpatiales**).

## 2.2 Importation d’images

### 2.2.1 Formats de données

* Format de type vision par ordinateur (RGB):
  + jpeg, png
* Format avec compression ou sans compression

Voici comment faire une liste

* Format RVB
  + jpeg
  + png
* GeoTiff
* COG
* NetCDF
* HDF5

#### 2.2.1.1 Formats de type RVB

Les premiers formats pour de l’imagerie à une bande (monochrome) et à trois bandes (image couleur rouge-vert-bleu) sont issus du domaine des sciences de l’ordinateur. On trouvera, entre autres, les formats pbm, png et jpeg. Ces formats supportent peu de métadonnées dans un entête (header) très limités. Ces formats restent très populaires dans le domaine de la vision par ordinateur.

#### 2.2.1.2 Le format GeoTiff

Le format GeoTIFF est une extension du format TIFF (Tagged Image File Format) qui permet d’incorporer des métadonnées géospatiales directement dans un fichier image. Développé initialement par Dr. Niles Ritter au Jet Propulsion Laboratory de la [NASA](https://www.earthdata.nasa.gov/esdis/esco/standards-and-practices/geotiff) dans les années 1990, GeoTIFF est devenu un standard de facto pour le stockage et l’échange d’images géoréférencées dans les domaines de la télédétection et des systèmes d’information géographique (SIG). Ce format supporte plus que trois bandes aussi longtemps que ces bandes sont de même dimension.

Le format GeoTIFF est très utilisé et est largement supporté par les bibliothèques et logiciels géospatiaux, notamment GDAL (Geospatial Data Abstraction Library), qui offre des capacités de lecture et d’écriture pour ce format. Cette compatibilité étendue a contribué à son adoption généralisée dans la communauté géospatiale.

##### 2.2.1.2.1 Standardisation par l’OGC

Le standard GeoTIFF proposé par l’Open Geospatial Consortium (OGC) en 2019 formalise et étend les spécifications originales du format GeoTIFF, offrant une norme robuste pour l’échange d’images géoréférencées. Cette standardisation, connue sous le nom d’OGC GeoTIFF 1.1 (2019), apporte plusieurs améliorations et clarifications importantes.

#### 2.2.1.3 Le format COG

Une innovation récente dans l’écosystème GeoTIFF est le format “Cloud Optimized GeoTIFF” (COG), conçu pour faciliter l’utilisation de fichiers GeoTIFF hébergés sur des serveurs web HTTP. Le COG permet aux utilisateurs et aux logiciels d’accéder à des parties spécifiques du fichier sans avoir à le télécharger entièrement, ce qui est particulièrement utile pour les applications basées sur le cloud.

### 2.2.2 Métadonnées des images

### 2.2.3 Données en géoscience

Calibration, unités, données manquantes, données éparses.

netcdf, xarray, GRIB.

Données météos, exemple avec SWOT.

## 2.3 Importation de données vectorielles

### 2.3.1 Importation d’un fichier *shapefile*

### 2.3.2 Importation d’une couche dans un *GeoPackage*

### 2.3.3 Importation d’une couche dans une *geodatabase* d’ESRI

### 2.3.4 Importation d’un fichier *shapefile*

## 2.4 Manipulation d’images

### 2.4.1 Manipulation d’une matrice

* Numpy

### 2.4.2 Mosaïquage, masquage et découpage

### 2.4.3 Changement de projection cartographique

### 2.4.4 Recalage d’images et co-registration

### 2.4.5 Requêtes attributaires

## 2.5 Manipulation de données vectorielles

## 2.6 Quiz de révision du chapitre

## 2.7 Exercices de révision

**Exercice 1.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.1.1](#sec-08011).

**Exercice 2.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.1.2](#sec-08012).

**Exercice 3.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.1.3](#sec-08013).

# 3. Réhaussement et visualisation d’images

Dans le chapitre, nous abordons…

**Liste des *packages* utilisés dans ce chapitre**

* Pour importer et manipuler des fichiers géographiques :
  + sf pour importer et manipuler des données vectorielles.
  + terra pour importer et manipuler des données matricielles.
* Pour construire des cartes et des graphiques :
  + tmap est certainement le meilleur *package* pour la cartographie.
  + ggplot2 pour construire des graphiques.

## 3.1 Visualisation sur le Web

WMS

## 3.2 Réhaussement visuel

Calcul d’histogrammes, étirement, égalisation, styling

## 3.3 Composés couleurs

* Vraies couleurs
* Fausses couleurs

## 3.4 Visualisation 3D

drapper une image satellite sur un DEM

## 3.5 Quiz de révision du chapitre

## 3.6 Exercices de révision

**Exercice 1.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.2.1](#sec-08021).

**Exercice 2.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.2.2](#sec-08022).

**Exercice 3.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.2.3](#sec-08023).

# 4. Transformations spectrales

Dans le chapitre, nous abordons les méthodes…

**Liste des *packages* utilisés dans ce chapitre**

* Pour importer et manipuler des fichiers géographiques :
  + sf pour importer et manipuler des données vectorielles.
  + terra pour importer et manipuler des données matricielles.
* Pour construire des cartes et des graphiques :
  + tmap est certainement le meilleur *package* pour la cartographie.
  + ggplot2 pour construire des graphiques.

## 4.1 Indices spectraux

## 4.2 Réduction de dimension

### 4.2.1 Analyses en composantes principales

## 4.3 Exercices de révision

**Exercice 1.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.3.1](#sec-08031).

**Exercice 2.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.3.2](#sec-08032).

**Exercice 3.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.3.3](#sec-08033).

# 5. Transformations spatiales

Dans le chapitre, nous abordons les méthodes…

**Liste des *packages* utilisés dans ce chapitre**

* Pour importer et manipuler des fichiers géographiques :
  + sf pour importer et manipuler des données vectorielles.
  + terra pour importer et manipuler des données matricielles.
* Pour construire des cartes et des graphiques :
  + tmap est certainement le meilleur *package* pour la cartographie.
  + ggplot2 pour construire des graphiques.

## 5.1 Analyse fréquentielle

## 5.2 Filtrage d’image

Filtre médian, filtre moyen

Filtre de Sobel, filtre Prewitt

## 5.3 Segmentation

## 5.4 Vectorisation et rasterisation

## 5.5 Analyse de terrain

### 5.5.1 Élévation

### 5.5.2 Pente

### 5.5.3 Ombrage

### 5.5.4 Visibilité

## 5.6 Quiz de révision du chapitre

## 5.7 Exercices de révision

**Exercice 1.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.4.1](#sec-08041).

**Exercice 2.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.4.2](#sec-08042).

**Exercice 3.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.4.3](#sec-08043).

# 6. Classifications d’images supervisées

Dans le chapitre, nous abordons les classifications supervisées….

**Liste des *packages* utilisés dans ce chapitre**

* Pour importer et manipuler des fichiers géographiques :
  + sf pour importer et manipuler des données vectorielles.
  + terra pour importer et manipuler des données matricielles.
* Pour construire des cartes et des graphiques :
  + tmap est certainement le meilleur *package* pour la cartographie.
  + ggplot2 pour construire des graphiques.

## 6.1 Classification d’images pixel par pixel

### 6.1.1 Parallélépipède

### 6.1.2 Méthodes paramétriques

### 6.1.3 Méthodes non paramétriques

### 6.1.4 SVEM, réseaux de neurones, forêts aléatoires

## 6.2 Segmentation d’images

### 6.2.1 Classification objet

### 6.2.2 Approches par arbre (BPT, etc.)

## 6.3 Quiz de révision du chapitre

## 6.4 Exercices de révision

**Exercice 1.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.6.1](#sec-08061).

**Exercice 2.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.6.2](#sec-08062).

**Exercice 3.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.6.3](#sec-08063).

# 7. Classifications d’images non supervisées

Dans le chapitre, nous abordons les classifications non supervisées….

**Liste des *packages* utilisés dans ce chapitre**

* Pour importer et manipuler des fichiers géographiques :
  + sf pour importer et manipuler des données vectorielles.
  + terra pour importer et manipuler des données matricielles.
* Pour construire des cartes et des graphiques :
  + tmap est certainement le meilleur *package* pour la cartographie.
  + ggplot2 pour construire des graphiques.

## 7.1 Classifications strictes

### 7.1.1 K-means

### 7.1.2 K-mediodes

### 7.1.3 Isodata

## 7.2 Classifications floues

## 7.3 C-Means

### 7.3.1 C-Means intégrant une dimension spatiale

## 7.4 Quiz de révision du chapitre

## 7.5 Exercices de révision

**Exercice 1.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.5.1](#sec-08051).

**Exercice 2.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.5.2](#sec-08052).

**Exercice 3.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.5.3](#sec-08053).

# 8. Introduction aux plateformes de mégadonnées

Dans le chapitre, nous abordons…

**Liste des *packages* utilisés dans ce chapitre**

* Pour importer et manipuler des fichiers géographiques :
  + sf pour importer et manipuler des données vectorielles.
  + terra pour importer et manipuler des données matricielles.
* Pour construire des cartes et des graphiques :
  + tmap est certainement le meilleur *package* pour la cartographie.
  + ggplot2 pour construire des graphiques.

## 8.1 Données massives

## 8.2 Manipulation de données satellitaires avec *Google Earth Engine*

## 8.3 Quiz de révision du chapitre

## 8.4 Exercices de révision

**Exercice 1.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.7.1](#sec-08071).

**Exercice 2.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.7.2](#sec-08072).

**Exercice 3.** À compléter

library(sf)  
library(terra)  
# ...  
# à compléter

Correction à la [section 9.7.3](#sec-08073).

# 9. Correction des exercices

## 9.1 Exercices du chapitre 1

### 9.1.1 Exercice 1

library(sf)  
library(terra)

### 9.1.2 Exercice 2

library(sf)  
library(terra)

### 9.1.3 Exercice 3

library(sf)  
library(terra)

## 9.2 Exercices du chapitre 2

### 9.2.1 Exercice 1

|  |
| --- |
| Figure 9.1: Exercice sur la contiguïté et les ordres d’adjacence |

### 9.2.2 Exercice 2

library(sf)  
library(terra)

### 9.2.3 Exercice 3

library(sf)  
library(terra)

Les valeurs du *I* de Moran sont les suivantes : 0,69 pour la matrice *Rook*, 0,52 pour la matrice inverse de la distance au carré réduite et 0,75 pour la matrice selon le critère des deux plus proches voisins.

## 9.3 Exercices du chapitre 3

### 9.3.1 Exercice 1

library(sf)  
library(terra)

### 9.3.2 Exercice 2

library(sf)  
library(terra)

### 9.3.3 Exercice 3

library(sf)  
library(terra)

## 9.4 Exercices du chapitre 4

### 9.4.1 Exercice 1

library(sf)  
library(terra)

### 9.4.2 Exercice 2

library(sf)  
library(terra)

### 9.4.3 Exercice 3

library(sf)  
library(terra)

## 9.5 Exercices du chapitre 5

### 9.5.1 Exercice 1

library(sf)  
library(terra)

### 9.5.2 Exercice 2

## Construction du réseau  
library(sf)  
library(terra)

### 9.5.3 Exercice 3

library(sf)  
library(terra)

## 9.6 Exercices du chapitre 6

### 9.6.1 Exercice 1

library(sf)  
library(terra)

### 9.6.2 Exercice 2

library(sf)  
library(terra)

### 9.6.3 Exercice 3

library(sf)  
library(terra)

## 9.7 Exercices du chapitre 7

### 9.7.1 Exercice 1

library(sf)  
library(terra)

### 9.7.2 Exercice 2

library(sf)  
library(terra)

### 9.7.3 Exercice 3

library(sf)  
library(terra)

# Bibliographie

Bonn, Ferdinand et Guy Rochon. 1992. *Précis de télédétection. Volume 1 : principes et méthodes*. Presses de l’université du Québec; Agence universitaire de la Francophonie.

Mather, Paul M et Magaly Koch. 2022. *Computer processing of remotely-sensed images*. John Wiley & Sons.

OGC. 2019. « OGC GeoTIFF Standard. » <https://docs.ogc.org/is/19-008r4/19-008r4.html/>.

Richards, John A, John A Richards et al. 2022. *Remote sensing digital image analysis*. Vol. 5. Springer.