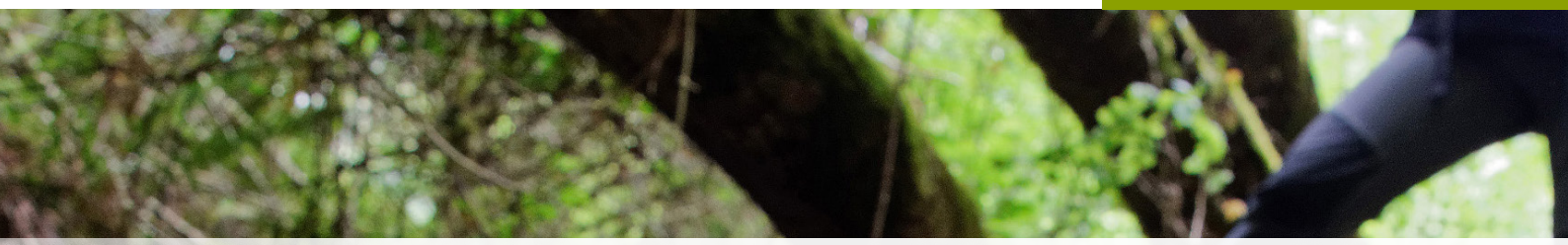
naturae

2022 • 2



Mise en forme et contrôle qualité de données, l'informatique au service de l'écologie

Florèn HUGON, Xavier NAVARRO, Matt RODRIGUEZ, Gabriel DAS NEVES, Matthieu BERRONEAU, Frank D'AMICO

art. 2022 (2) — Publié le 26 janvier 2022
www.revue-naturae.fr

OFB OFFICE FRANÇAIS DE LA BIODIVERSITÉ

PUBLICATIONS SCIENTIFIQUES

MUSÉUM NAL HIST NATURELLE

DIRECTEUR DE LA PUBLICATION/*PUBLICATION DIRECTOR*: Bruno David,
Président du Muséum national d'Histoire naturelle

RÉDACTEUR EN CHEF/*EDITOR-IN-CHIEF*: Jean-Philippe Siblet

ASSISTANTE DE RÉDACTION/*ASSISTANT EDITOR*: Sarah Figuet (naturae@mnhn.fr)

MISE EN PAGE/*PAGE LAYOUT*: Sarah Figuet

COUVERTURE/*COVER*:

Comptage des têtards de Grenouille des Pyrénées *Rana pyrenaica* Serra-Cobo, 1993 dans le cadre du protocole de suivi de la reproduction de l'espèce.
Crédit photo: Matthieu Berroneau.

Naturae est une revue en flux continu publiée par les Publications scientifiques du Muséum, Paris
Naturae *is a fast track journal published by the Museum Science Press, Paris*

Les Publications scientifiques du Muséum publient aussi/*The Museum Science Press also publish:*
Adansonia, Zoosystema, Anthropozoologica, European Journal of Taxonomy, Geodiversitas, **Cryptogamie sous-sections** ***Algologie, Bryologie, Mycologie, Comptes Rendus Palevol.***

Diffusion – Publications scientifiques Muséum national d'Histoire naturelle
CP 41 – 57 rue Cuvier F-75231 Paris cedex 05 (France)
Tél.: 33 (0)1 40 79 48 05/Fax: 33 (0)1 40 79 38 40
diff.pub@mnhn.fr/https://sciencepress.mnhn.fr

ISSN (électronique/*electronic*): 1638-9387

Mise en forme et contrôle qualité de données, l'informatique au service de l'écologie

Florèn HUGON
Université Pau et Pays Adour, E2S UPPA,
Laboratoire de Mathématiques et de leurs Applications de Pau (LMAP), UMR CNRS 5142,
UFR des Sciences et Techniques de la Côte basque,
Allée du Parc Montaury, F-64600 Anglet (France)
floren.hugon@univ-pau.fr

Xavier NAVARRO
Université Pau et Pays Adour, E2S UPPA, Laboratoire d'Informatique (LIUPPA),
UFR des Sciences et Techniques de la Côte basque,
Allée du Parc Montaury, F-64600 Anglet (France)
xavier.navarro@univ-pau.fr

Matt RODRIGUEZ
Gabriel DAS NEVES
Université Pau et Pays Adour, E2S UPPA,
UFR des Sciences et Techniques de la Côte basque,
Allée du Parc Montaury, F-64600 Anglet (France)
matt.rodriguez@etud.univ-pau.fr
gabriel.das-neves-rodrigues@etud.univ-pau.fr

Matthieu BERRONEAU
Cistude Nature, Chemin du Moulinat, F-33185 Le Haillan (France)
matthieu.berroneau@cisture.org

Frank D'AMICO
Université Pau et Pays Adour, E2S UPPA,
Laboratoire de Mathématiques et de leurs Applications de Pau (LMAP), UMR CNRS 5142,
UFR des Sciences et Techniques de la Côte basque,
Allée du Parc Montaury, F-64600 Anglet (France)
frank.damico@univ-pau.fr

Soumis le 19 novembre 2020 | Accepté le 10 mars 2021 | Publié le 26 janvier 2022

Hugon F., Navarro X., Rodriguez M., Das Neves G., Berroneau M. & D'Amico F. 2022. — Mise en forme et contrôle qualité de données, l'informatique au service de l'écologie. *Naturae* 2022 (2): 17-30. https://doi.org/10.5852/naturae2022a2

RÉSUMÉ
Dans de nombreuses disciplines scientifiques, des études expérimentales ou des suivis sur le terrain assurent la récolte de données. Celles-ci sont stockées sur des fichiers bruts avec un format intuitif, permettant une saisie facile par l'expérimentateur. Cependant, ce format brut est rarement directement compatible avec l'analyse des données récoltées et peut engendrer des analyses erronées ; il est nécessaire d'effectuer une mise en forme et un contrôle qualité des données. Face au nombre de jeux de données brutes croissants et toujours plus massifs, la discipline du numérique pour les sciences du vivant s'est développée. La programmation informatique représente une aide précieuse

pour les modélisateurs puisqu'elle permet d'automatiser la mise en forme et le contrôle qualité qui nécessitent souvent un nettoyage des données. Dans cet article, nous présentons une collaboration entre informaticien et modélisateur dans le cadre du suivi de l'abondance d'espèces animales. Les données récoltées sur plusieurs feuilles d'un tableur sont regroupées sur une seule et leur qualité est vérifiée. Les diverses fonctionnalités du programme effectuant cette vérification ont été mises en place à l'aide de la méthode « agile », méthode de développement informatique constituée de *sprints*. Après la fourniture d'une version du programme, un nouveau *sprint* définit une nouvelle fonctionnalité à mettre en place par l'informaticien dans une nouvelle version du programme. La première version permet l'appropriation du jeu de données par l'informaticien grâce à la fonctionnalité de mise en forme. Une version plus avancée gère l'absence de données, puis d'autres contrôlent la qualité des données récoltées et rapportent le traitement des anomalies détectées – donnée absente ou erronée ou en dehors d'une plage spécifiée – dans un fichier texte. Ce programme informatique a été explicité afin qu'il puisse être ré-approprié et ré-utilisé. Sa version complète est déposée dans GitHub. Le lien est donné en conclusion.

MOTS CLÉS
Abondance, gestion de données, méthode « agile », nettoyage des données, programmation informatique.

ABSTRACT
Data formatting and quality control, computing for ecology.
In many scientific disciplines, experimental studies or field monitoring ensure the data collection. The data is stored on raw files with an intuitive format, easily entered by the experimenter. However, this raw format is rarely directly compatible with the analysis of the collected data and the data may be erroneous; it is then necessary to carry out a formatting and quality control. Faced with the increasing number of ever more massive raw data sets, the discipline of digital for life sciences has developed. Computer programming represents a precious help for modellers since it allows the automation of data formatting and data cleaning. The data formatting makes it possible to obtain a format that will be directly used in the analyses. Automation also enables to avoid errors generated by manual formatting, such as typing errors and omissions. Data cleaning, a term used in computer science, corresponds to the data quality control according to the criteria provided by the modeller. The modeller knows the range of values of the data obtained and the possible errors produced. In this article, we present a collaboration between computer scientist and modeller in the framework of animal abundance monitoring. The data collected on several sheets of a spreadsheet had to be gathered on one sheet and their quality had to be checked. The various functionalities of the program carrying out this verification were implemented using the "agile" method, a computer development method consisting of sprints. After a version was provided, a new sprint defined the next functionality to be implemented by the computer scientist in a new version of the program. The first version allows the appropriation of the dataset by the computer scientist thanks to the formatting functionality. A more advanced version manages the absence of data, then others check the collected data quality and reports the processing of detected anomalies, data missing or erroneous or outside a specified range, in a text file. This computer program has been explained so that it can be re-appropriated and re-used, the full version is deposited in GitHub. The link is given in conclusion.

KEY WORDS
Abundance, data management, agile software development, data cleaning, computer programming.

INTRODUCTION

Très souvent dans le domaine des sciences, des expériences et/ou observations sont réalisées en laboratoire ou sur le terrain et les données sont stockées dans un fichier brut (voir définition en Annexe 1). Ce fichier, intuitif et explicite pour l'expérimentateur (voir définition en Annexe 1), lui permet de saisir les données récoltées facilement. Cependant, pour ensuite réaliser l'analyse des données, il est nécessaire de les mettre en forme en un fichier formaté nettoyé (voir définition en Annexe 1) qui répond aux besoins de l'analyse (De la Vega *et al.* 2019). Également, il est important d'effectuer un contrôle qualité des données afin que les erreurs de saisie soient minimisées et qu'elles aient un impact réduit sur les résultats de l'analyse (Van den Broeck *et al.* 2005). Ce contrôle s'effectue selon un processus appelé nettoyage des données ; il est décomposé en trois étapes. La première consiste à visualiser les données récoltées pour détecter les erreurs éventuelles. La seconde permet de faire le diagnostic des erreurs en les définissant. Enfin, la troisième permet d'éditer les erreurs selon des règles définies (Van den Broeck *et al.* 2005). Ces étapes de formatage et de nettoyage des données sont récurrentes dans les analyses (Singh & Gulwani 2016) et très chronophages (Kandel *et al.* 2011). Les jeux de données en écologie sont de plus en plus nombreux et importants, ainsi, automatiser leur mise en forme et leur nettoyage paraît tout à fait pertinent voire indispensable pour améliorer l'efficacité de l'analyse (Michener & Jones 2012 ; Carey *et al.* 2019).

De nombreux programmes informatiques (voir définition en Annexe 1) permettant le nettoyage des données sont déjà développés dans différents langages tels que Python, Java

ou encore Visual Basic d'Excel (Vaidya *et al.* 2011 ; Bosch *et al.* 2013 ; Singh & Gulwani 2016). Par exemple, dans le domaine de la génétique, SequenceMatrix© est un logiciel permettant de faciliter la lecture et l'analyse des séquences génétiques. Il permet de transférer l'information dans une feuille de calcul et de vérifier la qualité des séquences en ajoutant des informations telles que la longueur de la séquence, le nombre d'insertions, de délétions et des informations sur les codons (Vaidya *et al.* 2011). Dans d'autres domaines, il est possible d'utiliser NADEEF© ou encore Wrangler©, des plateformes interactives dans lesquelles l'utilisateur peut spécifier des règles de qualité des données et obtenir de l'aide (Kandel *et al.* 2011 ; Ebaid *et al.* 2013). Enfin, la plateforme Talend gère, comme sources d'informations, des données structurées mais son utilisation nécessite des compétences informatiques de description de la représentation des données. Elle permet de gérer des bases de données et présente des fonctionnalités de mise et forme et de contrôle qualité (Bowen 2012). D'autres outils sont aussi développés pour permettre l'homogénéisation des fichiers de données provenant de sources différentes (Rahm & Hai Do 2000). Ces outils permettent de stocker les données selon un format commun et ainsi de faciliter leur analyse. Cependant, ces différents outils informatiques requièrent la prise en main de leurs fonctionnalités. Il est crucial de bien comprendre les exécutions réalisées par une commande telle qu'une ligne de code ou un clic-bouton par exemple. De cette manière, il est parfois plus efficace d'écrire un nouveau code spécifique aux traitements à réaliser plutôt que d'utiliser un outil dont le fonctionnement sous-jacent n'est pas bien maîtrisé.

La discipline de l'informatique au service des sciences est en plein essor et de nombreuses formations universitaires sont désormais proposées pour répondre à cette demande croissante des sciences expérimentales (Valle & Berdanier 2012 ; Carey *et al.* 2019). L'élaboration d'un programme informatique est une source d'enrichissement pour l'informaticien et le modélisateur (voir définition en Annexe 1 ; Bosch *et al.* 2013) ; sa mise en œuvre offrant un gain temporel précieux dans les analyses, il est ainsi de plus en plus commun de mettre en place des collaborations entre les laboratoires d'écologie et d'informatique. La programmation informatique tente de devenir accessible dans de nombreux domaines pour remplacer partiellement ce qui était réalisé « à la main » mais elle intervient toujours en complément d'un travail empirique (Vee 2013). Dans ce processus, seul le modélisateur peut définir objectivement les critères de qualité des données selon leurs caractéristiques attendues, notamment en vue de leur traitement statistique ou mathématique, et une vraie collaboration s'installe entre les deux domaines.

Dans le contexte du changement climatique et de l'étude des réponses de la biodiversité, de nombreux suivis d'espèces sont réalisés, notamment pour modéliser l'abondance ou l'occupation au cours du temps (Mallard 2019). Dans le cadre de ces études, comme toujours en écologie où l'expérience est associée à une incertitude, l'observation des individus lors des sessions de suivi sur le terrain est soumise à ce que l'on appelle l'imperfection de la détection (Pellet & Schmidt 2005). Matérialisée sous forme d'une probabilité de détection, celle-ci peut être quantifiée grâce à un suivi réplicatif sur plusieurs sites et avec plusieurs visites (MacKenzie *et al.* 2002, 2003 ; Royle & Nichols 2003). Au cours de chaque visite, pour une analyse d'abondance, les individus observés sont comptabilisés (on parle de données de comptage) alors que pour une analyse d'occupation, seule la détection ou la non-détection (il s'agit d'une donnée binaire codée 0 ou 1) de l'espèce suivie est notée (Zipkin *et al.* 2017). Sachant que la détection peut être imparfaite, il est important de distinguer les termes présence/absence et détection/non-détection (Ferguson *et al.* 2015). Pour appréhender et intégrer l'hétérogénéité spatio-temporelle du suivi résultant de la réplication des sites et des visites, les conditions prévalentes lors des différentes visites sont notées sous forme de covariables dites d'échantillonnage (par exemple la température de l'air ou la nébulosité) tandis que les sites d'étude sont quant à eux caractérisés par des covariables dites de site (par exemple la couverture végétale ou l'altitude) (Miller *et al.* 2015 ; Bötsch *et al.* 2019). Ainsi, cette stratégie d'échantillonnage engendre une quantité importante d'informations à mettre en forme et à vérifier.

Concernant la modélisation de l'abondance ou de l'occupation avec des suivis réplicatifs de ce type, les méthodes disponibles dans le package *unmarked* pour le logiciel R (Fiske & Chandler 2011) et communément utilisées requièrent un tableau de données dans lequel chaque ligne représente un site d'étude et chaque colonne correspond à une variable collectée, soit un format « nettoyé » d'après Wickham (2014, concept de « tidy data »). Les méthodologies de nettoyage des données présentées précédemment semblent inadaptées à la mise en forme et au nettoyage des données de terrain, souvent récoltées dans un format peu conventionnel, par exemple, un fichier tableur avec des cellules fusionnées, plusieurs feuilles de données dans un même fichier ou encore différentes unités pour une même variable. D'après nos recherches bibliographiques, il n'existe pas de méthodologie simplifiée et vulgarisée à l'interface des domaines de l'écologie et de l'informatique qui permettent le traitement de données récoltées de cette manière. Ainsi, pour combler ce vide relatif, nous proposons ici des outils méthodologiques pour produire un fichier tableur de données formatées à partir de données brutes présentes dans différentes feuilles d'un autre fichier tableur. Après l'étape de mise en forme des données, nous effectuons leur contrôle qualité via plusieurs étapes de nettoyage. Nous illustrons notre méthodologie avec un cas d'étude, la modélisation de l'abondance de la Grenouille des Pyrénées (*Rana pyrenaica* Serra-Cobo, 1993) (Serra-Cobo 1993) dans le cadre d'un programme de recherche utilisant des protocoles standardisés. La présentation de la collaboration informaticien – modélisateur et celle d'une partie des codes informatiques a pour objectif de faciliter la mise en place de programmes informatiques pour le traitement de jeux de données conséquents en la démystifiant et de permettre la ré-appropriation de la méthode et des codes par une personne non-informaticienne.

MATÉRIEL ET MÉTHODES

Présentation du cas d'étude

La Grenouille des Pyrénées (*Rana Pyrenaica* Serra-Cobo, 1993) est l'une des espèces indicatrices du changement climatique étudiées dans le cadre du programme Les Sentinelles du Climat (Mallard & Couderchet 2019). Les suivis de la plupart des espèces indicatrices ont été réalisés annuellement de 2017 à 2021 afin de modéliser l'évolution de leurs abondances. Pour la Grenouille des Pyrénées, une des espèces d'amphibiens les plus rares d'Europe (Berroneau 2014), le suivi s'effectue par comptage des têtards le long des torrents dans lesquels l'espèce se reproduit. Chaque torrent étudié est découpé en différentes placettes indépendantes qui correspondent à des sites, pour conserver la cohérence avec les définitions de Royle & Nichols (2003). Au total, sept torrents sont suivis, découpés en quatre à 28 sites d'étude, tous visités trois fois au cours de la période de reproduction annuelle. Pour chaque torrent et chaque année, un fichier tableur permet la récolte des données de tous les sites. Pour chaque site, deux feuilles sont remplies, l'une avec les covariables d'échantillonnage, l'autre avec les covariables de site (Berroneau 2012). Ainsi, pour un torrent découpé en 11 sites d'étude, le tableur contient 2 × 11 feuilles de récolte de données. Sur l'ensemble des torrents étudiés, tous localisés dans le massif pyrénéen, 97 sites sont suivis, soit 194 feuilles de données remplies par année. Bien que le protocole ait été commun au cours des cinq années, différents observateurs se sont succédés chaque année, générant des différences dans la saisie des données entre les années. Compte tenu de la quantité et de la complexité des données récoltées dans ces fichiers bruts, il a paru utile de mettre en place un projet de programmation informatique. L'automatisation de cette mise en forme permet la modélisation de l'abondance de l'espèce en tenant compte des covariables de site et d'échantillonnage, non intégrées dans les premières analyses (Berroneau *et al.* 2015 ; D'Amico *et al.* 2019) sachant que la prise en compte de l'hétérogénéité spatio-temporelle est essentielle pour expliquer les variations de la probabilité de détection et de l'abondance (Zhao & Royle 2019).

Cadre de la programmation informatique

Le programme informatique a pour objectif de regrouper sur une seule feuille d'un fichier tableur (fichier formaté nettoyé, *output* [voir définition en Annexe 1]), les données récoltées sur plusieurs feuilles d'un autre fichier tableur (fichier brut, *input* [voir définition en Annexe 1]) tout en vérifiant leur qualité. L'objectif, est rappelons-le, d'obtenir un fichier formaté nettoyé, avec en ligne chaque observation et en colonne chaque variable (Wickham 2014). La première étape est la mise en forme des données. Elle nécessite l'écriture d'un code informatique qui recherche les données dans les différentes feuilles du fichier brut et les stocke dans l'unique feuille du fichier formaté ; cela constitue un premier *sprint* (voir définition en Annexe 1). La sélection des données d'intérêt à stocker est établie par le modélisateur selon sa question biologique et ses hypothèses. Toutes les données des fichiers bruts n'ont pas été stockées dans le fichier formaté. Par exemple, la date de la visite n'a pas été retenue dans cette étude car le modélisateur n'a pas jugé cette donnée pertinente pour l'explication de l'abondance ou de la probabilité de détection des têtards. La seconde étape correspond au contrôle qualité des données via plusieurs sous-étapes de nettoyage définies lors de chaque nouveau « sprint » ; entre autres, la détection et l'édition des données absentes et/ou erronées, i.e. les données qui ne respectent pas les critères de qualité.

Élaboration d'un cahier des charges

Un cahier des charges (voir définition en Annexe 1) fourni par le modélisateur permet de présenter le contexte de l'étude, de définir les objectifs et les livrables. Il indique notamment la sélection des données à mettre en forme et leur nature (nombre entier, nombre décimal, chaîne de caractères, etc.). Bien que le protocole expérimental soit aussi transmis, l'informaticien connaît peu les contraintes de terrain et les données qui peuvent être collectées, contrairement au modélisateur et à l'expérimentateur. Il est ainsi important de spécifier ces détails dans un document. Par exemple, une variable « nombre d'individus observés » est un nombre entier. L'écriture de ce document permet aussi de prendre conscience de l'hétérogénéité des données récoltées et met en lumière la nécessité de les traduire, notamment pour des variables collectées selon une chaîne de caractères. Dans notre cas d'étude, des variables codées avec les modalités « oui » ou « non » ont été traduites sous la forme 1 ou 0. Cela permet d'obtenir un jeu de données nettoyé homogène, plus facilement analysable. À la suite de la mise en forme des données, qui aura mis en lumière diverses anomalies suite aux tests du programme lors de l'élaboration de sa première version (voir définition en Annexe 1), le modélisateur pourra identifier les anomalies qu'il souhaite traiter automatiquement dans un nouveau *sprint.* Un second document est rédigé pour spécifier les critères de qualité de chaque variable et la gestion des données absentes et/ou erronées. Par exemple, il est rédigé « si la donnée est absente, écrire NA dans la cellule correspondante ». Ce document relatant les critères de qualité peut être partiellement écrit en amont de la livraison de la première version des fichiers formatés. Le modélisateur peut indiquer de manière générale le traitement des données absentes et les critères de qualité relatifs aux ordres de grandeur (variable quantitative) ou aux facteurs (variable qualitative) des variables mais n'est pas en mesure d'expliciter le traitement des diverses subtilités qui seront observées à l'issue de la première étape de mise en forme. Le cahier des charges peut également comporter une clause de confidentialité des données transmises. Ainsi, il est initialement léger et est alimenté au cours des nouveaux *sprints* mis en place. Lors de l'obtention de la version finale du programme, il reflète le déroulement de la collaboration.

Méthode « agile »

La méthode « agile » est une méthode de développement itérative selon laquelle les nouvelles fonctionnalités d'un programme informatique sont implémentées au fur et à

mesure via différents *sprints*, produisant des versions de plus en plus abouties. Cette méthode permet une grande efficacité dans l'écriture des programmes et une importante adaptabilité aux besoins du commanditaire (voir définition en Annexe 1 ; Cornic 2020). Dans notre cas d'étude, le premier *sprint* a réalisé la mise en forme des données en considérant qu'elles sont toutes correctes. Cette mise en forme a permis de visualiser les différentes anomalies communes dans les données : l'absence de données, traitée dans la deuxième version, et le non-respect de l'ordre de grandeur ou des facteurs de la variable, traité dans d'autres versions. L'édition d'un fichier texte relatant le traitement des anomalies détectées a également été implémenté. Ce fichier, essentiel pour vérifier la bonne exécution du programme, relate l'identification de l'anomalie (nom de la variable, emplacements dans le fichier brut et dans le fichier formaté), sa nature (absence, ordre de grandeur/facteur) et son édition éventuelle selon les directives données par le modélisateur. Pour tester ces versions, il peut être intéressant de produire un fichier test avec les anomalies de nature « absence » ou « non-respect de l'ordre de grandeur/facteur » sur toutes les variables. Enfin, la gestion des anomalies particulières a été effectuée dans une dernière version du programme, encore évolutive, selon les anomalies observées au cours de la mise en forme de l'ensemble des fichiers. Le programme informatique a été transmis avec ses différentes versions afin que le modélisateur s'approprie chaque étape de traitement des données et le code associé ; ceci lui offrant la possibilité d'éventuellement modifier le code dans une autre version. L'informaticien a également fourni au modélisateur un document d'aide à l'exécution du programme. Le programme a été écrit en langage Visual Basic en utilisant le logiciel Excel. La logique mise en place pourra être réutilisée dans d'autres langages.

RÉSULTATS

Organisation générale du programme informatique

L'exécution du programme sur un fichier brut permet de produire un fichier formaté et un fichier texte relatant le traitement des anomalies rencontrées. Le programme définit tout d'abord le type des variables nécessaires pour lire et écrire dans un fichier, c'est à dire les indices des boucles mises en place, les objets pour les noms des fichiers *input* et *output*, les chemins d'accès des fichiers et les types des variables métier (entier, décimal, etc. ; voir définition en Annexe 1). Ensuite, il effectue la demande du fichier *input* à traiter, son ouverture, la création et l'ouverture d'un fichier tableur *output* et d'un fichier texte. Puis, il remplit la première ligne du fichier *output* avec les noms des colonnes, soit les noms des variables métier et calcule le nombre de sites à traiter. Il exécute ensuite une boucle sur le nombre de sites, ce qui permet de traiter les données de chaque variable métier d'un même site puis d'effectuer le même traitement sur les données du site suivant. À la fin de chaque traitement d'un site, un message indique l'avancée du traitement. À la fin du traitement de toutes les données du site, les fichiers *input*, *output* et texte sont enregistrés et fermés.

Tableau 1. — Temporalité de l'élaboration du programme informatique.

Sprint	Temps dédié	Description
1	2 jours	Copie des données du fichier brut dans le fichier formaté, permet de détecter et lister des anomalies par le test du programme
2	1/2 jour	Vérification de la présence de la donnée
3	1/2 jour	Vérification des ordres de grandeurs des variables quantitatives
4	1/2 jour	Traduction des variables qualitatives en variable quantitative
5	1/2 jour	Vérification de la qualité de certaines variables par rapport à d'autres, largeur maximale > largeur minimale, somme des pourcentages égale à 100, etc.
6	1/2 jour	Traitement spécifique à certaines variables, agrégation de données brutes en une unique variable pour les variables relatives aux présences d'espèces prédatrices, gestion des unités, etc.

Temporalité de la collaboration informaticien – modélisateur pour l'élaboration du programme informatique

Pour chacune des 34 variables étudiées dans notre cas d'étude, le programme informatique recherche d'abord la donnée dans le fichier brut, puis vérifie que la cellule contient une information – donnée présente ou absente, puis si la donnée est présente, il vérifie sa qualité selon les critères notifiés dans le cahier des charges. Si la donnée est absente ou erronée, le programme remplit la cellule correspondante dans le fichier formaté selon les directives transmises dans le cahier des charges. Dans le cas présenté, le cahier des charges indique de remplir la cellule par NA lorsque la donnée est absente et d'écrire la donnée en rouge lorsque celle-ci est erronée. NA a été choisi car le programme R, sur lequel est réalisée l'estimation de l'abondance de l'espèce, reconnaît cette chaîne de caractères comme une donnée absente. Le rouge a été choisi pour faciliter la visualisation des erreurs dans les fichiers de sortie et leur édition – soit automatique lors d'un nouveau *sprint* si elles sont récurrentes, soit manuelle si elles sont anecdotiques. La mise en forme des données est associée à un premier sprint (la recherche de la donnée et sa copie dans le fichier formaté) et conduit à l'écriture de la première version du programme. Le contrôle qualité des données est assuré par de nombreux *sprints* supplémentaires. La vérification de la présence de la donnée correspond au *sprint* 2. Les nettoyages concernant les ordres de grandeur et la traduction des variables qualitatives ont été effectués dans les *sprints* 3 et 4. Enfin, les vérifications concernant plusieurs variables et la gestion des unités ont été réalisées dans deux autres *sprints* pour écrire la sixième version du programme. Les *sprints* de nettoyage ont été définis au fur et à mesure des visualisations des fichiers *output*. La temporalité des différents *sprints* est explicitée dans le Tableau 1.

	Visite	1	2	3
	Observateur	Prénom Nom	Prénom Nom	Prénom Nom
	Date	JJ/MM/AAAA	JJ/MM/AAAA	JJ/MM/AAAA
	Tair	15,7	17,1	18,3
	Teau	13,3	13,5	16,1
	Nébulosité	2	2	1
	Pluviométrie	non	non	non
	Débit	faible	faible	faible
	Nombre têtards	236	-120	32
	Nombre adultes	0	0	0
	Nombre pontes	0	0	0
	Nombre P.leniuculus	1	0	0
	Salmo trutta, P/A	A	A	A
	Natrix maura, P/A	A	A	A
	Commentaires			

	Visite	1	2	3
	Observateur	Prénom Nom	Prénom Nom	Prénom Nom
	Date	JJ/MM/AAAA	JJ/MM/AAAA	JJ/MM/AAAA
	Tair	17,2	17	18,5
	Teau	13,2	13,5	16
	Nébulosité	3	4	3
	Pluviométrie	non	non	non
	Débit	faible	moyen	faible
	Nombre têtards	1	0	
	Nombre adultes	0	0	0
	Nombre pontes	0	0	0
	Nombre P.leniuculus	0	0	0
	Salmo trutta, P/A	A	P	A
	Natrix maura, P/A	A	A	A
	Commentaires			

FIG. 1. — Extrait du fichier de données brutes stockées dans plusieurs feuilles d'un même fichier tableur pour un torrent découpé en quatre sites d'étude. Mise en évidence de la variable « nombre de têtards observés », comptabilisés à chaque visite sur les sites 1 (**onglet A1**) et 2 (**onglet A2**). Les onglets A stockent le nombre d'observation et les covariables d'échantillonnage ; les onglets B, les covariables de site.

	ObsV1	ObsV2	ObsV3	TairV1	TairV2	TairV3
Site1	236	-120	32	15,7	17,1	18,3
Site2	1	0	NA	17,2	17	18,5
Site3	5	10	4	16,8	15,8	18,5
Site4	18	19	5	17,3	17,2	18,2

FIG. 2. — Extrait du fichier formaté, un tableur avec en ligne les sites d'étude et en colonne les variables étudiées. Les suffixes V1, V2, V3 indiquent la visite relative à chaque donnée. La valeur –120 ne respecte pas le critère de qualité de la variable « nombre de têtards observés », elle est donc indiquée en rouge. Il n'y avait pas de données de comptage pour le site 2 à la visite 3, la valeur indiquée est donc NA.

```
fichier origine : Nom_du_fichier_input_brut.xlsx
fichier de destination : Nom_du_fichier_ouput_formate.xlsx

---------- Site 1----------

Erreur : Variable ObsV2 (type Integer) (Position d'origine : Cellule D10
-> Position de destination : Cellule C2) du site 1 (Feuille A1) : -120
    -> Conditions de cohérence :  la valeur est un entier positif
        -> Remplissage effectué : incoherence donc -120 en rouge gras


---------- Site 2----------

Erreur : Variable ObsV3 (type Integer) (Position d'origine : Cellule E10
-> Position de destination : Cellule D3) du site 2 (Feuille A2) : vide
    -> Conditions de cohérence :  la valeur est un entier positif
        -> Remplissage effectué : donnee absente donc NA
```

FIG. 3. — Extrait du fichier texte rendant compte des anomalies rencontrées.

CODE INFORMATIQUE POUR LE TRAITEMENT D'UNE VARIABLE COLLECTÉE SUR UN SITE

Des extraits des données brutes sont présentés en Figure 1, des données formatées en Figure 2 et du fichier texte en Figure 3. La première version du programme (lignes de code bleues dans la Figure 4) effectue le copier-coller des informations contenues dans plusieurs feuilles d'un tableur (Fig. 1), dans une seule feuille d'un autre tableur (Fig. 2). La seconde version (ajout des lignes de code violettes dans la Figure 4) permet la détection des données absentes et leur gestion selon les directives du modélisateur. Dans l'exemple présenté, la donnée est absente dans la cellule E10 de la feuille A2 du fichier *input* brut (Fig. 1) et il est donc indiqué NA dans la cellule D3 du fichier *output* formaté (Fig. 2). Les autres versions (ajout des lignes de code rouges dans la Figure 4) permettent la détection des anomalies et leur édition selon les critères de qualité indiqués. Dans l'exemple présenté, pour la variable « nombre de têtards observés », le critère est « la valeur est un entier positif », la cellule D10 de la feuille A1 du fichier brut contient une valeur négative, cette valeur est donc notée en rouge dans la cellule C2 du fichier formaté (Figs 1, 2). Enfin, l'exécution des fonctions *Erreurs_Anomalie* et *Action_Anomalie* (ajout des lignes de code bordeaux dans la Figure 4, Annexe 2) permet l'édition du fichier texte relatant toutes les anomalies rencontrées – donnée absente ou erronées – et leur édition (Fig. 3). Le code pour la gestion des données absentes est commun à toutes les variables contrairement au code pour la gestion de la qualité de la donnée, qui est spécifique du critère de qualité de chaque variable. Les codes présentés sont construits de façon à être les plus facilement compréhensibles par des personnes non initiées, qui peuvent ainsi facilement identifier et vérifier les parties de code organisées en différents *sprints* pour chaque variable. Le programme complet est disponible à l'adresse https://github.com/FloHugon/DataFormatting, dernière consultation le 6 janvier 2022.

QUELQUES CAS PARTICULIERS

Les critères de qualité diffèrent selon les variables étudiées, ils peuvent concerner une seule variable, plusieurs variables en même temps, une agrégation de variables ou encore une traduction de variable. La qualité des covariables de site représentant les pourcentages d'occupation du sol peut être

vérifiée pour chaque variable (valeur entre 0 et 100) mais aussi sur la somme de ces variables qui doit être égale à 100 (Annexe 3). La détection/non-détection d'une espèce prédatrice peut constituer une covariable d'échantillonnage dans certains relevés ; dans ce cas, il peut être pertinent d'agréger les résultats observés sur les n visites pour obtenir plutôt une covariable de site, qui indiquera la présence du prédateur sur le site. Ici, la présence d'une espèce prédatrice a été considérée si elle a été observée au moins une fois au cours des trois visites. Dans notre exemple, la Truite (*Salmo trutta* Linnaeus, 1758), prédatrice des œufs et des têtards de la Grenouille des Pyrénées, a été détectée lors de la visite 2 sur le site 2 (cellule D14 de la feuille A2 ; Fig. 1), l'agrégation a conduit à noter que l'espèce était présente sur le site d'étude (Annexe 4). Enfin, des variables sous forme d'une chaîne de caractères peuvent être traduites en une variable numérique qui pourra être incluse dans l'analyse. Par exemple, le débit noté faible, moyen ou fort peut être traduit par faible = 1, moyen = 2, fort = 3 (Annexe 5).

DISCUSSION

Utilisation de la méthode « agile », nettoyage manuel *versus* automatisé

Apport majeur de la démarche présentée ici, le programme fourni permet d'obtenir des fichiers formatés automatiquement à partir des fichiers bruts et de vérifier la qualité des données récoltées ainsi que de corriger les erreurs éventuelles via une étape d'édition des données. Son exécution permet de quasi-supprimer le temps alloué au formatage des tableaux pour l'estimation de l'abondance. Dans notre cas d'étude, le plus grand tableau formaté contenait 952 données : l'ensemble des 34 variables (nombre d'observations, covariables d'échantillonnage et de site) pour chacun des 28 sites d'un des torrents étudiés. Il est évident que sa mise en forme manuelle aurait été longue et fastidieuse, générant probablement des erreurs liées à des fautes de frappe ou une mauvaise lecture dans les 56 feuilles du fichier brut. Suite à la mise en forme, le contrôle qualité via diverses étapes de nettoyage a été réalisé au sein de nombreux *sprints*. La méthode « agile » a permis ici de développer successivement les processus de nettoyage, en les ajoutant un à un dans le code informatique. Manuellement, cela aurait nécessité d'effectuer un premier nettoyage pour l'absence des données puis un second pour les ordres de grandeurs, et ainsi de suite. Le modélisateur aurait été contraint de revoir l'ensemble des données plusieurs fois, chaque étape d'édition pouvant générer une nouvelle fois des erreurs. Il aurait également pu essayer d'effectuer plusieurs nettoyages en une seule étape, ce qui aurait probablement généré des erreurs étant donné le nombre de données à traiter ; au total ici 34 variables × 97 sites répartis sur sept torrents, soit 3298 données. Les fichiers obtenus à l'issue de la dernière version fournie requièrent une vérification des données et le traitement des données erronées, signalées en rouge. Ce traitement peut être manuel ou constituer un nouveau *sprint* dans le programme informatique. La procédure

```
' Traitement de la variable métier, ObsV1 = Nombre de têtards observés à la visite 1
' Détection cellule vide équivalent à l'absence de la donnée
If Workbooks(fichierOrigine).Worksheets(nomOngletA).Range("C10").Value = "" Then
  ' Traitement de l'absence de données
  Workbooks(fichierDest).Worksheets("Feuil1").Cells(i + 1, 2) = NA
  'Compte-rendu de l'absence de données
  Erreurs_Anomalie Tberror, "ObsV1", "Integer", "C10", "B" + CStr(i + 1),
  CStr(i),
  nomOngletA, "vide"
  Action_Anomalie Tberror, "la valeur est un entier positif", "donnee absente
  donc NA"
  Else
  ' Récupération de la donnée
  ObsV1 = Workbooks(fichierOrigine).Worksheets(nomOnglet-
  tA).Range("C10").Value
  ' Détection anomalie
  If ObsV1 < 0 Then
  ' Traitement anomalie : mise en rouge de la cellule
  Workbooks(fichierDest).Worksheets("Feuil1").Range("B" + CStr(i +
  1)).Font.Color =
  RGB(255, 0, 0)
  ' Compte-rendu de l'anomalie
  Erreurs_Anomalie Tberror, "ObsV1", "Integer", "C10", "B" + CStr(i + 1),
  CStr(i),
  nomOngletA, CStr(ObsV1)
  Action_Anomalie Tberror, "la valeur est un entier positif", CStr(ObsV1)
  End If
  Workbooks(fichierDest).Worksheets("Feuil1").Cells(i + 1, 2) = ObsV1
End If
```

Fig. 4. — Extrait d'une partie du code informatique pour le traitement d'une variable, ici le nombre d'observation à la visite 1 (ObsV1). Les lignes bleues sont écrites lors de la première version afin de récupérer la donnée. Les lignes violettes/rouges sont ajoutées à l'issue de la seconde/troisième version pour détecter et gérer les données absentes / les anomalies. Enfin, les lignes bordeaux permettent l'édition d'un fichier qui rend compte du traitement.

étape par étape qu'offre la méthode « agile » permet d'affiner les besoins de nettoyage au fur et à mesure de l'étude dans des *sprints* successifs, tout en désacralisant la fausse idée de la complexité du code informatique, qui pourrait effrayer certains modélisateurs et expérimentateurs pour la mise en place d'une telle collaboration. La facilité d'utilisation de cette méthode réside dans l'écriture d'un cahier des charges succinct, le découpage d'un projet en *sprints* précis et une évolution du code informatique sans limite, par intégration de nouveaux morceaux de code sans modification des codes précédents. Elle confère ainsi une grande flexibilité du code informatique et une compréhension importante par des personnes non-initiées.

Une infinité de traitements possibles grâce à la programmation informatique

Les exemples de code informatique présentés ici ne sont qu'une partie des codes écrits. Néanmoins, ils représentent déjà différents traitements classiquement effectués sur des données récoltées en écologie, soit la vérification de la présence de la donnée, de l'ordre de grandeur – variable positive et incluse dans un certain intervalle ou encore d'une somme de pourcentages. Nous avons aussi abordé des cas plus précis, le traitement d'une variable avec unité – donnée brute en mètres ou en centimètres, l'agrégation de données et la comparaison de variables. Une infinité de

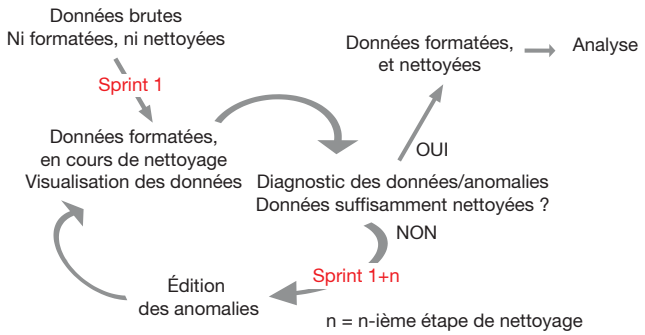


Fig. 5. — Déroulement de la collaboration modélisateur – informaticien pour l'écriture d'un programme informatique

traitements existe, propre à chaque variable et à l'objectif de l'étude. Par exemple, il est aussi possible de mettre en place la conversion de données de comptage en données de détection/non-détection. Les données de comptage sont récoltées dans le cadre de la modélisation d'abondance, mais parfois, les observations sont rares (ou difficiles à obtenir) et il est alors conseillé d'effectuer plutôt une modélisation de l'occupation, fondée sur des données binaires, 0 pour la non-détection et 1 pour la détection (Zipkin *et al.* 2017). Un programme peut toujours être amélioré, il est dynamique et malléable selon les souhaits d'analyse du modélisateur ; ceci est hautement facilité si l'implémentation a été réalisée avec la méthode « agile ». Les codes fournis illustrent dans différents contextes une démarche et une logique de traitement que chacun pourra s'approprier pour produire par mimétisme des codes informatiques adaptés à sa problématique. Une étroite collaboration entre le modélisateur et l'informaticien permettra ainsi d'ajouter de nombreuses fonctionnalités dans une version du programme toujours plus aboutie.

Un développement en Visual Basic pour plus d'accessibilité

Dans ce cas d'étude, le logiciel R n'a pas été choisi car il nous a semblé moins intuitif et moins abordable pour les raisons évoquées plus haut. Le modélisateur ne connaissait ni les outils sous R pour effectuer une mise en forme (gestion des feuilles de données, des cellules fusionnées, etc.), ni ceux pour nettoyer des données (package *tidyverse*, Wickham *et al.* 2019). Les données brutes étaient compilées dans un format de fichier Excel, l'informaticien était disponible et compétent en Visual Basic. Ce choix était donc le moins coûteux en temps et énergie. De plus, la plupart des structures effectuant des suivis sur le terrain sont familières de l'utilisation d'Excel et de Visual Basic contrairement à celle de R. Nous souhaitions proposer des méthodes abordables à la plus grande majorité de la communauté ciblée. Les packages du *tidyverse* étant nombreux et requiérant une bonne connaissance de R, cela aurait pu apporter trop de complexité. La programmation sous Visual Basic, a dans notre cas d'étude été très efficace, mais la logique de développement mise en œuvre pourrait tout à fait être transposée lors d'un traitement réalisé sous R.

Démarche de la collaboration modélisateur – informaticien

Comme évoqué en introduction, le nettoyage des données s'effectue en plusieurs étapes, visualisation des données, diagnostic des anomalies, édition des anomalies (Van den Broeck *et al.* 2005). La démarche que nous avons suivie lors de notre collaboration modélisateur – informaticien correspond à ces trois étapes. En effet, il a été d'abord nécessaire de scinder le projet en sous-projets, correspondant à différents *sprints* réalisés au cours du développement informatique. Le premier *sprint* a permis l'obtention du jeu de données formaté et la visualisation des données puis les suivants ont contribué au nettoyage progressif. Ici, un *sprint* a permis le nettoyage des données absentes puis d'autres ont réalisé le nettoyage des données erronées en termes d'ordre de grandeur, de facteurs, d'unité, de comparaison inter-variable, etc. Les étapes visualisation, diagnostic et édition forment une boucle ; celle-ci s'arrête au moment où le modélisateur évalue que les données sont suffisamment nettoyées. Les échanges réguliers dans un vocabulaire commun entre le modélisateur et l'informaticien, qui appartiennent tous deux à des champs disciplinaires très différents, sont essentiels pour assurer la progression continue et efficace des divers objectifs du projet (Carey *et al.* 2019) ; ils sont particulièrement nécessaires pour définir les *sprints* successifs. Par exemple, l'informaticien a exprimé le besoin d'un document relatant la nature des variables, leurs critères de qualité et les modalités de gestion des anomalies détectées. En revanche, le diagnostic des données, la décision des étapes de nettoyage et la définition des critères de qualité appartiennent au modélisateur, qui aura échangé avec l'expérimentateur pour bien connaître les modalités de collecte des données. La démarche conseillée, présentée dans la Figure 5, permet de répondre à toute problématique, étape par étape, en l'enrichissant si besoin au cours du développement.

Anticiper les analyses pour optimiser l'écriture du programme informatique

L'écriture d'un programme informatique peut être longue (De la Vega *et al.* 2019) mais ce coût temporel apporte le grand bénéfice d'un traitement homogène, reproductible et efficace des jeux de données brutes (Valle & Berdanier 2012). Dans le domaine de la génétique, pour permettre la ré-analyse efficace de données, des règles définissent le format commun des données selon lequel elles doivent être stockées dans les bases de données (Jurburg *et al.* 2020). Ainsi, pour gagner en efficacité, il pourrait être pertinent pour les futures analyses, que l'expérimentateur récolte les données dans des fichiers bruts semi-formatés présentant un format et une collecte de données homogène. Des règles précises de notation des données devraient être définies afin de réduire le temps alloué à l'écriture de codes pour le traitement d'anomalies particulières. Les variables qualitatives devraient, par exemple, être notées selon un choix de facteurs et les variables quantitatives collectées dans la même unité. Dans notre cas d'étude, le programme a été développé à partir des fichiers tableurs de l'année 2017. Cependant, en l'exécutant sur les fichiers des autres années, une erreur sur le nom des feuilles de données a été reportée,

et une autre est récurrente sur plusieurs variables. L'examen de quelques fichiers a révélé que des cellules avaient été fusionnées contrairement aux fichiers de 2017 et que les noms des feuilles étaient légèrement différents. Nous avons corrigé les formats des fichiers manuellement car c'était ici la solution la plus simple et la plus rapide. Anticiper en amont un format commun semi-formaté pour les fichiers de récolte de données aurait évité de perdre du temps à rechercher l'explication de l'erreur et à effectuer ces corrections. De manière plus générale, pour toute question scientifique, il est vivement conseillé de réaliser une planification expérimentale en définissant d'abord le type d'analyse nécessaire pour répondre à la question, puis le protocole expérimental adapté à l'exécution de cette analyse (Seltman 2012 ; Barker & Milivojevich 2016). La planification expérimentale est trop peu mise en œuvre alors qu'elle est essentielle pour une recherche efficace, reproductible et valorisable ; elle permet entre autres de ne pas générer d'erreurs évitables, ce qui est souvent moins coûteux que de corriger des données mal récoltées (Seltman 2012 ; Barker & Milivojevich 2016). Dans le cas de collecte de jeux de données importants, comme dans notre cas d'étude, la planification expérimentale peut être accompagnée d'un plan de gestion des données, élaboré en partenariat avec l'informaticien qui sera chargé du nettoyage de celles-ci. Cela permettra de définir, par exemple, les formats des données, les métadonnées nécessaires, ou encore les modalités de stockage à long terme des données (Michener & Jones 2012).

CONCLUSION

Le développement d'un programme informatique pour la mise en forme et le contrôle qualité des données a permis un gain temporel fort par rapport à l'effort manuel couramment mis en œuvre qui constitue environ 80 % du temps de l'analyse (Michener & Jones 2012). Le développement en Visual Basic a donné l'opportunité de visualiser rapidement l'ensemble des données collectées, initialement présentes dans de nombreux fichiers peu lisibles, afin de diagnostiquer les données et de définir les critères de qualité et les règles de nettoyage. La méthode « agile » utilisée a permis un développement progressif, facile et intuitif du programme informatique via les nombreux échanges entre informaticien et modélisateur. Elle est ici utilisée dans un développement en Visual Basic mais elle pourra également être mise en œuvre lors d'un nettoyage sur le logiciel R. Bien que la collaboration présentée soit celle de l'informaticien et du modélisateur, le programme informatique a été écrit par des étudiants de première année ; ceci témoigne de l'accessibilité de l'écriture d'un programme et contribue à démystifier l'informatique auprès des personnes non initiées. Le programme complet est disponible à l'adresse suivante https://github.com/FloHugon/DataFormatting (dernière consultation le 6 janvier 2022), afin de faciliter son appropriation. Notre expérience de collaboration a renforcé notre conviction de promouvoir les échanges entre les équipes qui font de la modélisation – en écologie, mais aussi dans d'autres champs disciplinaires et les équipes d'informaticiens. Dans un contexte de production de jeux de données toujours plus importants, nous encourageons vivement l'utilisation de l'informatique et la mise en place de collaborations pour assurer l'efficacité et la reproductibilité de la mise en forme et du contrôle qualité des données.

Remerciements
Nous remercions Cistude Nature et les partenaires du programme les Sentinelles du Climat qui nous ont autorisé à communiquer les données d'abondance aux étudiants informaticiens pour la réalisation de ce projet, ainsi que les différents financeurs de ce programme, l'Union européenne (FEDER), la région Nouvelle Aquitaine et les départements des Pyrénées-Atlantiques et de la Gironde. Nous soulignons l'investissement particulier de deux étudiants, Matt Rodriguez et Gabriel Das Neves, qui ont permis l'aboutissement de ce projet en produisant le programme informatique partiellement présenté ici. Nous remercions enfin les deux rapporteurs, Jocelyn Champagnon et Clément Calenge, pour leurs remarques nombreuses, constructives et détaillées qui nous ont permis d'améliorer grandement la présentation de cette collaboration.

RÉFÉRENCES

Barker T. B. & Milivojevich A. 2016. — *Quality by Experimental Design.* CRC Press, Boca Raton, 721 p.

Berroneau M. 2012. — *Protocole de caractérisation des populations de Grenouille des Pyrénées, programme pluriannuel de conservation (2012-2014) de la Grenouille des Pyrénées.* Cistude Nature, Le Haillan, 8 p.

Berroneau M. 2014. — *La Grenouille des Pyrénées, une endémique de l'ouest pyrénéen*. Cistude Nature, Le Haillan, 41 p.

Berroneau M., D'Amico F., Fournier A., Desvaux B. & Chazal R. 2015. — Trois années de suivi des populations françaises de *Rana pyrenaica* Serra-Cobo, 1993 (Amphibia: Ranidae) : premières estimations d'abondance des têtards. *Bulletin de la Société herpétologique de France* 156: 31-44.

Bosch N., D'Mello S. & Mills C. 2013. — What emotions do novices experience during their first computer programming learning session?, *in* Lane H. C., Yacef K., Mostow J. & Pavlik P. (éds), *Artificial Intelligence in Education* Vol. 7926. Springer, Berlin: 11-20. https://doi.org/10.1007/978-3-642-39112-5_2

Bötsch Y., Jenni L. & Kéry M. 2019. — Field evaluation of abundance estimates under binomial and multinomial N -mixture models. *Ibis* 162 (3): 902-910. https://doi.org/10.1111/ibi.12802

Bowen J. P. 2012. — *Getting Started with Talend Open Studio for Data Integration: Develop System Integrations with Speed and Quality Using Talend Open Studio for Data Integration.* Packt Publishing, Birmingham, 299 p.

Carey C. C., Ward N. K., Farrell K. J., Lofton M. E., Krinos A. I., McClure R. P., Subratie K. C., Figueiredo R. J., Doubek J. P., Hanson P. C., Papadopoulos P. & Arzberger P. 2019. — Enhancing collaboration between ecologists and computer scientists: lessons learned and recommendations forward. *Ecosphere* 10 (5). https://doi.org/10.1002/ecs2.2753

Cornic C. 2020. — *Gérer vos projets informatiques avec les méthodes agiles.* https://blog-gestion-de-projet.com/gerez-vos-projets-informatiques-avec-les-methodes-agiles/, dernière consultation le 18 janvier 2021.

D'Amico F., Berroneau M. & Lasserre V. 2019. — *Rana pyrenaica*

(Serra-Cobo 1993) – Grenouille des Pyrénées, *in* Mallard F. (éd.), *Programme les sentinelles du Climat. Tome VIII: écologie du changement climatique en région Nouvelle Aquitaine*. Cistude Nature, Le Haillan: 267-308.

De la Vega A., García-Saiz D., Zorrilla M. & Sánchez P. 2019. — Lavoisier: high-level selection and preparation of data for analysis, *in* Schewe K.-D. & Singh N. K. (éds), *Model and Data Engineering*. 9th International Conference, MEDI 2019, Toulouse, France, October 28-31, 2019. Springer International Publishing (coll. LNCS; 11815), Cham: 50-66. https://doi.org/10.1007/978-3-030-32065-2_4

Ebaid A., Elmagarmid A., Ilyas I.F., Ouzzani M., Quiane-Ruiz J.-A., Tang N. & Yin S. 2013. — NADEEF: a generalized data cleaning system. *Proceedings of the VLDB Endowment* 6 (12): 1218-1221. https://doi.org/10.14778/2536274.2536280

Ferguson P. F. B., Conroy M. J. & Hepinstall-Cymerman J. 2015. — Occupancy models for data with false positive and false negative errors and heterogeneity across sites and surveys, *in* Yoccoz N. (éd.), *Methods in Ecology and Evolution* 6 (12): 1395-1406. https://doi.org/10.1111/2041-210X.12442

Fiske I. & Chandler R. 2011. — Unmarked: an R Package for fitting hierarchical models of wildlife occurrence and abundance. *Journal of Statistical Software* 43 (10): 1-23. https://doi.org/10.18637/jss.v043.i10

Jurburg S. D., Konzack M., Eisenhauer N. & Heintz-Buschart A. 2020. — The archives are half-empty: an assessment of the availability of microbial community sequencing data. *Communications Biology* 3 (474): 1-8. https://doi.org/10.1038/s42003-020-01204-9

Kandel S., Paepcke A., Hellerstein J. & Heer J. 2011. — *Wrangler: Interactive Visual Specification of Data Transformation Scripts*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11). Association for Computing Machinery, New York: 3363-3372. https://doi.org/10.1145/1978942.1979444

MacKenzie D. I., Nichols J. D., Lachman G. B., Droege S., Andrew Royle J. & Langtimm C. A. 2002. — Estimating site occupancy rates when detection probabilities are less than one. *Ecology* 83 (8): 2248-2255. https://doi.org/10.1890/0012-9658(2002)083[2248:ESORWD]2.0.CO;2

MacKenzie D. I., Nichols J. D., Hines J. E., Knutson M. G. & Franklin A. B. 2003. — Estimating site occupancy, colonization, and local extinction when a species is detected imperfectly. *Ecology* 84 (8): 2200-2207. https://doi.org/10.1890/02-3090

Mallard F. 2019. — *Tome VIII : Écologie du changement climatique en région Nouvelle-Aquitaine, Programme les sentinelles du climat*. Cistude Nature, Le Haillan, 605 p.

Mallard F. & Couderchet L. 2019. — Climate Sentinels Research Program: developing indicators of the effects of climate change on biodiversity in the region of New Aquitaine (South West, France), *in* Leal Filho W., Barbir J. & Preziosi R. (éds), *Handbook of Climate Change and Biodiversity*. Springer International Publishing, Cham: 223-241. https://doi.org/10.1007/978-3-319-98681-4_14

Michener W. K. & Jones M. B. 2012. — Ecoinformatics: supporting ecology as a data-intensive science. *Trends in Ecology & Evolution* 27 (2): 85-93. https://doi.org/10.1016/j.tree.2011.11.016

Miller D. A. W., Bailey L. L., Grant E. H. C., McClintock B. T., Weir L. A. & Simons T. R. 2015. — Performance of species occurrence estimators when basic assumptions are not met: a test using field data where true occupancy status is known, *in* Gimenez O. (éd.), *Methods in Ecology and Evolution* 6 (5): 557-565. https://doi.org/10.1111/2041-210X.12342

Pellet J. & Schmidt B. R. 2005. — Monitoring distributions using call surveys: estimating site occupancy, detection probabilities and inferring absence. *Biological Conservation* 123 (1): 27-35. https://doi.org/10.1016/j.biocon.2004.10.005

Rahm E. & Hai Do H. 2000. — Data cleaning: problems and current approaches. *Bulletin of the Technical Committee on Data Engineering* 23 (04): 3-13

Royle J. A. & Nichols J. D. 2003. — Estimating abundance from repeated presence-absence data or point counts. *Ecology* 84 (3): 777-790. https://doi.org/10.1890/0012-9658(2003)084[0777:EAFRPA]2.0.CO;2

Seltman H. J. 2012. — *Experimental Design and Analysis*. Carnegie Mellon University, Pittsburgh, 428 p.

Serra-Cobo J. 1993. — Descripcion de una nueva especie europea de *Rana parda* (Amphibia, Anura, Ranidae). *Alytes* 11 (1): 1-15.

Singh R. & Gulwani S. 2016. — *Transforming spreadsheet Data Types Using Examples*. Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '16). Association for Computing Machinery, New York: 343-356. https://doi.org/10.1145/2837614.2837668

Vaidya G., Lohman D. J. & Meier R. 2011. — SequenceMatrix: concatenation software for the fast assembly of multi-gene datasets with character set and codon information. *Cladistics* 27 (2): 171-180. https://doi.org/10.1111/j.1096-0031.2010.00329.x

Valle D. & Berdanier A. 2012. — Computer programming skills for environmental sciences. *Bulletin of the Ecological Society of America* 93 (4): 373-389. https://doi.org/10.1890/0012-9623-93.4.373

Van den Broeck J., Argeseanu Cunningham S., Eeckels R. & Herbst K. 2005. — Data Cleaning: detecting, diagnosing, and editing data abnormalities. *PLoS Medicine* 2 (10): e267. https://doi.org/10.1371/journal.pmed.0020267

Vee A. 2013. — Understanding computer programming as a literacy. *Literacy in Composition Studies* 1 (2): 42-64. https://doi.org/10.21623/1.1.2.4

Wickham H. 2014. — Tidy data. *Journal of Statistical Software* 59 (10): 1-23. https://doi.org/10.18637/jss.v059.i10

Wickham H., Averick M., Bryan J., Chang W., McGowan L., François R., Grolemund G., Hayes A., Henry L., Hester J., Kuhn M., Pedersen T., Miller E., Bache S., Müller K., Ooms J., Robinson D., Seidel D., Spinu V., Takahashi K., Vaughan D., Wilke C., Woo K. & Yutani H. 2019. — Welcome to the Tidyverse. *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686

Zhao Q. & Royle J. A. 2019. — Dynamic N-mixture models with temporal variability in detection probability. *Ecological Modelling* 393: 20-24. https://doi.org/10.1016/j.ecolmodel.2018.12.007

Zipkin E. F., Rossman S., Yackulic C. B., Wiens J. D., Thorson J. T., Davis R. J. & Grant E. H. C. 2017. — Integrating count and detection-nondetection data to model population dynamics. *Ecology* 98 (6): 1640-1650. https://doi.org/10.1002/ecy.1831

Soumis le 19 novembre 2020 ;
accepté le 10 mars 2021 ;
publié le 26 janvier 2022.

ANNEXES

ANNEXE 1. — Glossaire.

Termes	Définition
Cahier des charges	Ensemble de documents rédigés par le modélisateur à destination de l'informaticien pour la mise en œuvre du programme
Commanditaire	Celui qui contacte l'informaticien pour répondre à un besoin
Expérimentateur/ Observateur	Celui qui collecte les données et les notifie dans le fichier brut
Fichier brut	Fichier rempli par l'expérimentateur, entré en *input* dans le programme informatique
Fichier formaté nettoyé	Fichier obtenu en *output* du programme informatique
Informaticien	Celui qui écrit le programme informatique
Input	Ce qui est mis en entrée dans un programme informatique
Modélisateur	Celui qui souhaite analyser les données du fichier brut (ici, il s'agit du commanditaire)
Output	Ce qui est obtenu en sortie du programme informatique
Programme informatique	Séquence d'opérations à effectuer, souvent destinées à être exécutées par un ordinateur sous la forme d'un algorithme
Sprint	Itération de développement brève avec un objectif précis et succinct
Variable Métier	Variable ayant une identification venant du métier (ici l'écologie)
Version	Produit obtenu à l'issue de l'ajout d'un *sprint* dans le programme

Annexe 2. — Code des fonctions Erreurs_Anomalie et Action_Anomalie qui éditent le rapport d'erreur.

```
Erreurs_Anomalie (Tberror() As String, nomVariable As String, typeVariable As String,
PositionOrigine As String, PositionDestination As String, site As String, feuille As String,
remplissage As String)
Dim error As String
-- Explique l'anomalie, la donnée concernée, son type, sa position dans les deux fichiers, le site sur lequel elle a été récoltée,
le remplissage effectué --
error = «Erreur  : Variable « + nomVariable + «(type « + typeVariable + «) (Position
d'origine : Cellule « + PositionOrigine + « -> Position de destination : Cellule « +
PositionDestination + «) du site « + CStr(site) + « (Feuille « + feuille + «) : « +
remplissage
Tberror(i + 1) = error
Print #1, Tberror(i + 1)


Action_Anomalie(Tberror() As String, incoherence As String, remplissage As String)
Dim error As String
-- Gestion de l'anomalie « donnée manquante », indique que NA est noté en noir --
If remplissage = «donnee absente donc NA» Then
error = « -> Conditions de cohérence : « + incoherence
Tberror(i + 1) = error
Print #1, Tberror(i + 1)
error = « -> Remplissage effectué : « + remplissage
Tberror(i + 1) = error
Print #1, Tberror(i + 1)
error = «»
Tberror(i + 1) = error
Print #1, Tberror(i + 1)
Else
-- Gestion de l'anomalie « donnée erronée », indique que la valeur est notée en rouge --
error = « -> Conditions de cohérence : « + incoherence
Tberror(i + 1) = error
Print #1, Tberror(i + 1)
error = « -> Remplissage effectué : « + remplissage + « en rouge»
Tberror(i + 1) = error
Print #1, Tberror(i + 1)
error = «»
Tberror(i + 1) = error
Print #1, Tberror(i + 1)
End If
```

Annexe 3. — Contrôle qualité faisant intervenir plusieurs variables.

```
-- Traitement des variables des strates végétatives en pourcentage, Mousse, Herbacée, Buissonnante et Supérieure --
If (Mousse + Herb + Buiss + Sup) <> 100 Then
' Traitement anomalie, si la somme est différente de 100 alors les 4 valeurs sont écrites en rouge
Workbooks(fichierDest).Worksheets(«Feuil1»).Range(«AF» + CStr(i + 1)).Font.Color =
 RGB(255, 0,0)
Workbooks(fichierDest).Worksheets(«Feuil1»).Range(«AG» + CStr(i + 1)).Font.Color =
RGB(255, 0,0)
Workbooks(fichierDest).Worksheets(«Feuil1»).Range(«AH» + CStr(i + 1)).Font.Color =
 RGB(255, 0,0)
Workbooks(fichierDest).Worksheets(«Feuil1»).Range(«AI» + CStr(i + 1)).Font.Color =
RGB(255, 0,0)
-- Compte rendu de l'anomalie --
...
End If
```

Annexe 4. — Agrégation de plusieurs variables.

```
-- Traitement des variables métier, Strutta1, Strutta2 et Strutta3, les présences et absences de l'espèce Salmo trutta Linnaeus,
1758 au cours des visites 1, 2 et 3 --
StruttaOri1 = Workbooks(fichierOrigine).Worksheets(nomOngletA).Range(«C28»).Value
StruttaOri2 = Workbooks(fichierOrigine).Worksheets(nomOngletA).Range(«E28»).Value
StruttaOri3 = Workbooks(fichierOrigine).Worksheets(nomOngletA).Range(«G28»).Value
If StruttaOri1 = «A» And StruttaOri2 = «A» And StruttaOri3 = «A» Then
    StruttaDest = 0
Workbooks(fichierDest).Worksheets(«Feuil1»).Cells(i + 1, 29) = StruttaDest
Workbooks(fichierDest).Worksheets(«Feuil1»).Range(«AC» + CStr(i + 1)).Font.Color =
RGB(0, 0, 0)
ElseIf
StruttaOri1 = «P» Or StruttaOri2 = «P» Or StruttaOri3 = «P» Then
StruttaDest=1
Workbooks(fichierDest).Worksheets(«Feuil1»).Cells(i + 1, 29) = StruttaDest
Workbooks(fichierDest).Worksheets(«Feuil1»).Range(«AC» + CStr(i + 1)).Font.Color =
RGB(0, 0, 0)
Else
-- code pour gestion de l'anomalie si une information est manquante sur les trois visites --
…
End If
```

Annexe 5. — Traduction d'une chaîne de caractères vers une valeur numérique.

```
-- Traitement de la variable métier, DebitV1, le débit de l'eau à la visite 1 --
DebitV1Ori = Workbooks(fichierOrigine).Worksheets(nomOngletA).Range(«C9»).Value
If CStr(DebitV1Ori) = «» Then
… -- code pour gestion d'une donnée absente --
Else
Select Case DebitV1Ori
Case « faible », « Faible »
DebitV1Dest = 1
Workbooks(fichierDest).Worksheets(«Feuil1»).Cells(i + 1, 17) = DebitV1Dest
Workbooks(fichierDest).Worksheets(«Feuil1»).Range(«Q» + CStr(i + 1)).Font.Color =
RGB(0, 0, 0)
… -- code pour cases « moyen », « Moyen », « fort », « Fort » --
Case Else
… -- code pour gestion de l'anomalie si une autre chaîne de caractères est écrite --
End Select
End If
```