Renard – Lapins – Carottes : une modélisation de système proie-prédateur

Beaurepaire Louis – Delatte Thomas – Ghazi Virgile – Prat Adélie

ARE - DYNAMIC

Résumé : Le projet sur lequel nous travaillons s'intitule **"Equilibre d'un écosystème"**. Il consiste en la modélisation interdépendante de deux populations : proies et prédateurs. Plus particulièrement, on s'intéresse ici à l'évolution du nombre de renards et de lapins au sein d'un même milieu, l'objectif étant de déterminer si un état d'équilibre du rapport proie/prédateur est possible et sous quelles conditions il est atteint. Afin de répondre au mieux à cette problématique, plusieurs questions seront soulevées telles que **"Si un tel équilibre est possible alors pour quelles valeurs est-il obtenu ?"** ou encore **"Quelles sont les variables pertinentes à prendre en compte** pour **modéliser efficacement les interactions proies/prédateurs ?»**. Enfin, à l'issu de sa réalisation, une dernière question constituera l'analyse critique de notre modèle : **"Dans quelles mesures celui-ci est-il ressemblant à la réalité et quelles en sont les limites ?»**.

A la suite de l'étude voici les résultats marquants : une stabilité de la population émerge quelque soit les conditions initiales (tant qu'elles restent réalistes), les grosses meutes desservent la survie des agents. On remarque aussi que l'accélération de l'émergence de la stabilité est corrélée à un seuil de besoins assez bas.

Fondements

Il s'agit ici de rendre compte de notions fondamentales sur lesquelles repose notre travail. On pourrait se pencher en premier lieu sur les **deux agents** que nous faisons intervenir, éléments clés autour desquels se construit notre programme. Il est donc nécessaire de définir brièvement ce terme. De ce fait, un « agent » caractérise ce qui est à l'origine d'un processus ou d'un phénomène. Au sein de notre modélisation, les lapins et les renards endossent ce rôle : ils effectuent des déplacements et exécutent des actions selon leur état (s'ils ont faim ou envie de se reproduire). Ils déterminent donc les changements dans notre matrice espace de départ, en particulier la disparition de certains agents (lapins mangés ...) et l'apparition de nouveaux (reproduction). Un système d'**interaction** entre les agents est ainsi instauré, autre notion essentielle de notre travail.

Une notion très importante intervient ici afin de lier les deux agents, c'est celle d'interdépendance. Dans notre cas, c'est le rapport proie/ prédateur qui rend chacune des deux espèces dépendantes de l'autre. Cela rejoint bien tout l'enjeu de notre système à étudier l'équilibre d'un écosystème et voir si la cohabitation d'une population de proies et de prédateurs dans un milieu naturel est possible. Plus précisément, cette idée illustre le fait que le nombre de renards dépend de la quantité de ressources alimentaires disponible, soit le nombre de lapins. Réciproquement, la population de lapins diminue lorsque la population de prédateurs renards est élevée. Cette étude justifie l'ajout d'un agent « tiers », les carottes, qui représentent elle l'alimentation du lapin de laquelle dépend sa survie.

Par ailleurs, il est nécessaire d'aborder la **simplification de la vie réelle** que constitue notre programme, notion capitale lorsqu'on réalise une modélisation. En effet, au cours de la création d'un projet visant à la simulation d'un phénomène existant, on fait le choix de l'intervention de certains facteurs et, à l'inverse, on en néglige d'autres. Par suite, le rendu n'est pas une représentation parfaite de ce qui se passe en pratique. Graphiquement, cela s'illustrerait par le fait que sur une courbe sinusoïdale, on ne prendrait en considération que la ligne médiane, approximation moyenne des processus rencontrés. Le choix des variables entrant en jeu est donc capitale pour espérer obtenir un résultat le plus proche possible de la réalité. En ce qui concerne notre travail nous avons donc tenté d'employer les caractéristiques qui nous semblaient cruciaux pour la mise en modèle de la réalité. A titre d'exemple, nous pouvons mentionner en premier lieu la taille de la matrice qui pose le cadre de notre modélisation et le nombre de renards et de lapins (variable) que nous y faisons intervenir. La probabilité de présence de carottes intervient aussi, comme représentation de l'accès à la nourriture pour les lapins. On considère également l'espérance de vie, la maturité sexuelle ou encore le temps de gestation de chacune des deux espèces ainsi que l'efficacité/chance de réussite de la chasse du renard (sur la base de données réelles). Enfin, les seuils d'excitation et de faim de chaque animal -déterminant la façon dont il va agir : manger ou se reproduire- et le gain d'excitation et de faim résultant de chacune de ces interactions sont pris en compte. Bien que pourvu de nombreuses variables, cela s'apparente à une simplification de la vie réelle. En effet, ici on ne prend notamment pas en compte les aléas qui pourraient survenir dans la réalité tels que les maladies, les blessures entraînant la mort ainsi que les cas de fausse-couche, stérilité empêchant la reproduction, ou encore la présence d'autre animaux dans le régime alimentaire du renard. Là se situe donc tout l'intérêt de la vérification avec les données existantes et les travaux déjà menés.

Introduction

La question initiale nous guidant fut « Dans quelles conditions un équilibre entre une population de proies et une autre de prédateurs peut-il émerger ? ». Cette question dû être reformulée durant cette étude comme nous le verrons. La majorité de notre travail repose sur une modélisation par un modèle itératif, cette modélisation sera présentée plus amplement dans la suite de ce dossier. Notre groupe est constitué de quatre personnes, une annexe présente en fin de dossier les présente eux et leur rôle plus en détail.

Présentation du code et analyse :

La majeure partie du code consiste en des interactions entre deux bases de données : un dictionnaire *agents* et une matrice *environnement*. Ils se présentent chacun sous la forme suivante :

```
agents = {
    UUID() : ("type", (agent))
    UUID() : ("type", (agent))
    }
```

Ici *UUID()* est une clé unique d'identification, cette clé n'aura qu'un seul et unique élément associé. Le « *type* » indique si c'est un renard, un lapin, ou une carotte. Le deuxième attribut du *tuple* contient toutes les caractéristiques de notre *agent*, on se réfèrera notamment à l'annexe décrivant la construction de *l'agent* renard indiquant son contenu.

L'environnement se présente de la manière suivante :

```
environnement = array([[[UUID(), UUID()] ... [UUID()]]
                              .       .         .
                              .         .       .
                              .           .     .
                     [[UUID()],         ...    []  ]])
```

Une matrice de taille n*n de listes content des *UUID()*. L'idée du programme commence ainsi à germer. Les échanges entre *environnement* et *agents* permettent ainsi une mise à jour constante des deux malgré les itérations permanentes sur l'un des deux éléments. On reviendra plus tard cet avantage.

Voici la boucle principale de notre programme, elle contient en filiation toutes les autres fonctions et il suffit alors d'une exécution de fonction unique pour initialiser et dérouler la simulation :

```
#On init l'environnement
environnement = init_matrice()
#On init le dico des agents
agents = merge_two_dicts(creation_carottes(),merge_two_dicts(creation_lapins(), creation_renards()))
#On update la matrice avec les uuid du dico
update_matrice()
#On exécute n tours
tours(nb_jour)
```

On initialise l'environnement en créant une matrice de liste de la taille voulue. On crée ensuite le dictionnaire des agents en générant tous les agents nécessaires. On incorpore ensuite nos agents dans l'environnement grâce à *update_matrice()* qui va placer tous les *UUID()* dans la matrice. On exécute ensuite le nombre de tour voulu, tout en sachant qu'un tour correspond à une journée.

Ainsi on va s'atteler ici à détailler comment se déroule un tour :

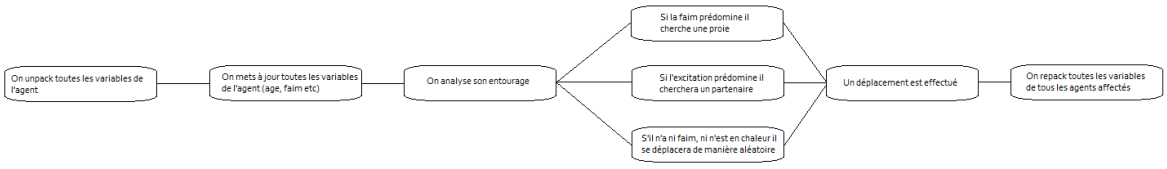
```
for a in agents:
    type_agent, agent = agents[a] #on unpack notre agent et son type
    if type_agent == "lapin": #on teste le type de notre agent
        _, _, _, _, _, _, _, position = agent #on unpack la position de notre agent
        ligne, colonne = position
        if a in environnement[ligne, colonne]:
            #on teste si notre agent est dans l'environnement
            #si le test échoue cela correspond à un agent mort
            #tous les agents ayant existés sont dans le dictionnaire
            #seuls les vivants sont dans l'environnement
            mouvement_lapin(a)

    elif type_agent == "renard":
        _, _, _, _, _, _, _, position = agent
        ligne, colonne = position
        if a in environnement[ligne, colonne]:
            mouvement_renard(a)

    else:
        _, position = agent
        ligne, colonne = position
        if a in environnement[ligne, colonne]:
            mouvement_carotte(a)
```

Un tour se déroule de la façon suivante, on itère un *for* dans le dictionnaire des *agents*. On vérifie son type afin de savoir quel type de mouvement il doit suivre. On vérifie ensuite si l'agent est toujours vivant ou non. Dans le cas où il est toujours actif on le fait effectuer le mouvement correspondant à son type.

On ne détaillera ici que le type de mouvement du renard et cela pour plusieurs raisons dont une volonté de concision, un mouvement de lapin très similaire, et un mouvement de carotte inintéressant. Voici donc schématisé le déplacement du renard :



Etape n°1 : on *unpack* toutes les variables de notre *agent* concerné.

```
_, renard = agents[id_renard]
age, sexe, efficacite, fecondite, excitation, faim, gestationnaire, position = renard
ligne, colonne = position
en_gestation, jour, nb = gestationnaire
```

Etape n°2 : on met à jour ses variables, cela correspond à rajouter un jour à son *age*, vérifier s'il doit mourir de vieillesse, augmenter sa *faim* et son *excitation*, et vérifier s'il ne doit pas accoucher.

```
#On met à jour ses variables
age += 1 #il prend un jour de plus
if age > esperance_vie_renard:
    environnement[ligne, colonne].remove(id_renard)
    return
#le sexe ne change pas
#l'efficacite ne change pasu
#la fecondite ne change pas
excitation += gain_excitation_renard
faim += gain_faim_renard
if faim > 1:
    environnement[ligne, colonne].remove(id_renard)
    return
#on s'occupe du tuple gestationnaire
if en_gestation:
    jour += 1
    if jour == temps_gestation_renard:
        cpt_renardeaux = 0
        while cpt_renardeaux < nb:
            cpt_renardeaux += 1
            id_enfant, enfant = creation_renard(1,(ligne, colonne))
            agents[id_enfant] = enfant
            environnement[ligne, colonne].append(id_enfant)
        print "%s renardeaux sont nés" % nb
        en_gestation = False
        jour = 0
        nb = 0
        gestationnaire = (en_gestation, jour, nb)
#La position ne change pas
```

Etape n°3 : on analyse son environnement, on en crée des *listes de UUID* à proximité directe pour savoir si des proies ou partenaires sont présents dans son champ de vision.

```
entourage_moore = [] #liste des uuid aux environs

for a in range((ligne-M),(ligne+M)+1):
    for b in range((colonne-M),(colonne+M)+1):
        for c in environnement[a%n,b%n]:
            if c != id_renard:
                entourage_moore.append(c)

#On retire les carottes de l'entourage
entourage_moore_renard = []
entourage_moore_lapin = []

for e in entourage_moore:
    type_agent,_ = agents[e]
    if type_agent == "renard":
        entourage_moore_renard.append(e)
    elif type_agent == "lapin":
        entourage_moore_lapin.append(e)
```

Etape n°4 : le renard décide s'il a envie de manger, de se reproduire, ou de se déplacer aléatoirement. Le code ne sera pas mis ici car trop conséquent et avec un intérêt très limité. Cependant on peut préciser que sa décision se fera en fonction de ses niveaux *d'excitation* et de *faim* et des *agents* à proximité.

Etape n°5 : on repack alors tous les *agents* dont les valeurs ont été modifiées (un lapin mort par exemple, ou un renard n'ayant plus faim).

Remarque : on peut alors se repencher sur les avantages de ce système de dialogue entre la matrice environnement et le dictionnaire agents. Pour un tour on itère sur tout le dictionnaire agents, impossible donc de faire une naissance sans que *Python* ne détecte une erreur. Ce système permet alors d'éviter ça, les erreurs d'itération sont évitées.

Analyser les matrices immenses s'avère impossible, on a donc créé un rendu graphique sous la forme d'une animation rendant compte des déplacements des renards et des lapins en indiquant le nombre d'agent du type présent sur la case. Des liens menant à des vidéos sont disponibles dans le README.md sur le GitHub.

A partir de là nous avons pu commencer à analyser notre modèle et la portée qu'il avait. En regardant des dizaines de simulation de plusieurs années on constate assez rapidement que les populations stagnent assez rapidement, i.e. qu'elles ne bougent plus. On peut alors avancer deux théories pour expliquer cela :

- *Cas n°1 :* un bug s'est glissé dans le programme et certains agents échappent à la mort et aux déplacements. Si on peut d'abord penser à cela par le fait qu'ils ne meurent pas cette théorie est rapidement mise en échec par les (trop) nombreuses relectures du code à la recherche d'une quelconque raison menant à cette rébellion de certains agents.
- *Cas n°2 :* notre modèle, même s'il ne reflète pas une réalité concrète (quel renard survivrait sans bouger ?) réussit à installer un état d'équilibre où des renards et lapins survivent sans bouger. Plusieurs indices permettent de consolider cette hypothèse : le rapport entre le nombre de renards et de lapins résiduels immobiles est peu ou prou similaire à celui entre les renards et les lapins initiaux ; cet état d'équilibre ne survient pas pour des données absurdes ; cet « bug » est présent depuis le début, malgré des changements algorithmiques significatifs.

Dans l'hypothèse où nous sommes en présence du cas n°2 on peut faire plusieurs conclusions. Notamment qu'un état d'équilibre s'atteint presque inévitablement tant que les valeurs d'initialisation ne sont pas absurdes (un millier de renard ne trouvera pas d'équilibre avec un unique lapin). Ainsi notre question initiale « Peut-on atteindre un état d'équilibre, et quelles sont les valeurs le permettant ? » trouve sa réponse : « Toute valeur non absurde mène à son état d'équilibre ».

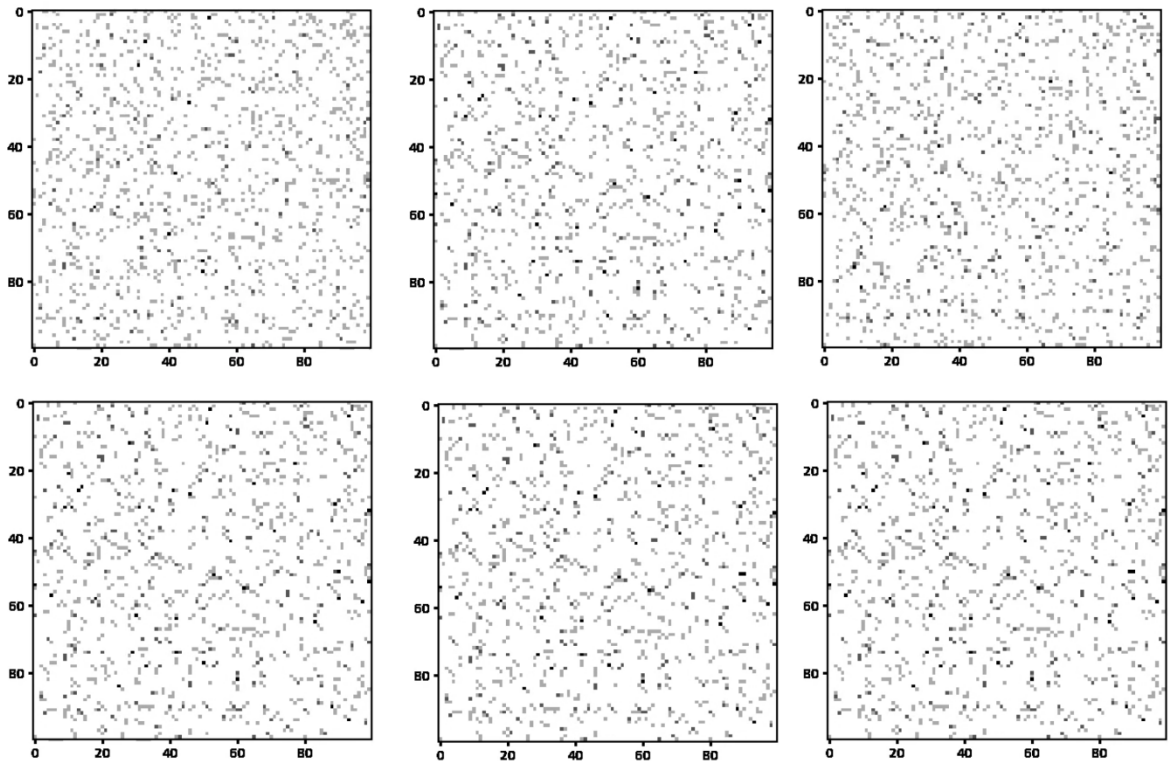
D'autres outils d'analyses ont été développés durant ce projet, la majorité sont trop peu présentables pour être introduits en détail ici, cependant l'on s'arrêtera sur l'outil qui donnait la moyenne du nombre d'agent par case occupées. Les résultats n'ont pas pu être présentés sous une forme convenable à temps mais le résultat suivant à pu être extrait des données brutes : les agents (renards ou lapins) n'ont pas de tendance particulière à se regrouper en meutes. On peut aller même plus loin : si des meutes de 20 ou 25 renards ou lapins se créent aléatoirement elles se défont presque immédiatement par des morts consécutives, on en déduit la chose suivante : « La meute imposante n'est pas profitable à la survie ».

Une autre expérience a été conduite, à partir de combien de temps voyait-on émerger une condition stable pour les agents. Il a en effet été établi que celle-ci survenait presque inévitablement, il nous a

donc semblé intéressant de reformuler notre question initiale « Peut-on atteindre un état d'équilibre, et quelles sont les valeurs le permettant ? » en « Quelles valeurs permettent d'atteindre le plus rapidement possible un état d'équilibre ? ». Encore une fois le code et les valeurs obtenues sont très archaïques et ne seront pas présentés ici. Aucune valeur précise n'a été établie, l'état stable n'ayant pas été défini de manière rigoureuse il aura été jugé « à l'œil ». Cependant on peut avancer sans douter la corrélation suivante : plus le seuil des besoins (incitant les animaux à avoir des mouvements contrôlés) est bas, et plus la stabilité arrivait vite. Nous n'avancerons cependant pas de lien de causalité direct par manque de données chiffrées précises (cela est du au manque de temps notre question principale ayant dû être reformulée suite aux résultats obtenus).

Dans un souci de fiabilité on précisera ici les conditions dans lesquels les tests énoncés ci-dessus ont été réalisés :

- Chaque test a été effectué une dizaine de fois, dès qu'un paramètre changeait le test été refait 10 fois
- La matrice faisait du 100*100
- La même machine a été utilisée pour faire les 10 tests
- Les tests avec des temps de calcul supérieurs à 15min était automatiquement rejetés et arrêtés



On a ci-dessus un exemple de l'affichage graphique utilisé : ici la simulation a été faite avec 1000 lapins et 300 renards. Les images représentent la population de lapins aux jours 3, 13, 25, 50, 73, et 100. On remarque une apparition de condition stable à partir du 70ème jour.

Conclusions

Virgile Ghazi

Au cours de ce semestre, notre groupe a programmé une simulation d'un environnement constitués de lapins, renards et carottes. J'ai, pour ce projet apporté une certaine contribution : tout d'abord, je suis celui à l'origine de l'idée de ce projet. J'ai ainsi, avec Louis Beaurepaire, décidé des grandes lignes de ce que notre simulation ferait et sous quel format elle se présenterait (ici, une animation d'un plan de 100*100) et ai programmé certaines fonctions comportementales des agents comme celles de reproduction ou de chasse. J'ai également participé dans la recherche de données, avec l'aide de Adélie Prat, ce qui nous a permis de fixer certaines valeurs concernant les attributs des différents agents. Je me suis également occupé de la partie affichage de la simulation. Cette partie étant particulièrement complexe, j'ai donc dû demander l'aide de Louis pour finir (et revoir en grande partie) cette étape.
Bien que n'ayant pas été le plus investi dans la programmation (Louis et Thomas ayant écrit la plupart du code), mais le programme m'ayant été expliqué à chaque étapes de son avancement, ce travail de groupe m'a permis d'apprendre beaucoup, notamment le fonctionnement et l'utilisation de matrices, comment faire une animation, le fonctionnement des UUIDs... Bien que n'ayant jamais fait de programmation avant cette année, j'ai pu apprendre beaucoup de choses que ce soit par mes camarades ou par les professeurs du cours d'ARE.

Adélie Prat

Tout au long de la réalisation de ce projet, mon rôle a été la recherche de données utiles au codage, la mise en page sur GitHub et le codage des sous-fonctions mineures. En conclusion, je pourrais retenir plusieurs éléments positifs que m'ont apportée ce module d'ARE et ce projet.
En premier lieu, j'ai pu encore davantage expérimenter le travail de groupe qui nécessite certaines compétences : savoir s'écouter les uns les autres, se répartir les rôles équitablement mais aussi composer avec les qualités et les défauts de chacun. Or, cela pourrait m'être utile si je suis amenée plus tard à travailler en collaboration. D'autre part, par le biais de ce travail, j'ai pu me familiariser avec l'écriture en Markdown ainsi qu'avec une plateforme en ligne que je ne connaissais pas telle que GitHub, très intéressante pour un projet de groupe puisqu'elle permet à chacun des membres de modifier et consulter les documents qui y sont sauvegardés. J'ajouterai à cela la gestion du temps, l'organisation et l'efficacité que le projet m'a enseignée puisqu'il n'a pas été facile de mener à bien le travail dans le temps imparti. Enfin, n'étant pas très à l'aise avec l'univers de la programmation, le module d'ARE m'a permis de m'y intéresser davantage et d'en comprendre plus précisément le fonctionnement.

Louis Beaurepaire

Tout au long de ce projet j'ai principalement dû endosser deux rôles : celui de chef d'équipe et celui de programmeur. Si l'association des deux menait à une charge de travail accrue on peut y trouver une certaine logique, celui qui a le nez directement dans le code est le plus à même de savoir comment aiguiller l'équipe, quels sont les problèmes et échéances à prévoir. Cependant, en posant un œil critique sur mon travail au cours du semestre je me rends compte que si j'ai su fournir un code efficace avec Thomas Delatte et Virgile Ghazi c'était au détriment d'une direction claire ; une difficulté m'est ainsi apparue dès le début, coordonner tout le monde alors que l'on est soi-même une ressource revient à s'imposer une rigueur dont je préfère me passer en temps normal. Ainsi si ce projet m'a apporté des connaissances en

programmation c'est surtout les compétences induites que je retiendrai : une rigueur accrue dans l'auto-gestion, comment gérer une équipe, qu'est-ce qu'un planning efficace ou encore de l'importance de s'accorder une zone tampon en fin de projet pour s'occuper des imprévus.

Au cours de ce projet sur lequel nous avons travaillé j'ai avec Louis Beaurepaire codé la simulation et plus particulièrement repérer les possibles problèmes rencontrés. J'ai aussi codé les programmes nécessaires pour l'analyse des données. Au cours de ce projet j'ai appris à utiliser différentes fonctionnalités de Jupyter Notebook et comment utiliser au mieux plusieurs outils, notamment les variables globales et les matrices. De même, nous avons appris comment modéliser au mieux un système complexe comme celui proie/prédateur grâce aux données et aux outils mis à notre disposition. J'ai été ravi de travailler sur un projet autant passionnant que complexe.

Thomas Delatte

Summary

The project we are working on is called "Ecosystem Balance". It consists of the interdependent modelling of two populations: preys and predators. More specifically, we are interested here in the evolution of the number of foxes and rabbits within the same environment, the objective being to determine whether a stable state of the prey/predator ratio is possible and under what conditions it is achieved. In order to answer this question, several interrogations will be raised such as "If such a balance is possible then for what values is it obtained?" or "What are the relevant variables to consider to effectively model prey/predator interactions? Finally, at the end of its realization, a last question will constitute the critical analysis of our model: "To what extent is it similar to reality and what are its limits? ".

Following the study here are the striking results: a stability of the populations emerges whatever the initial conditions were (as long as they remain realistic), the large packs penalize the survival of the agents. We also note that the acceleration in the emergence of stability is correlated with a fairly low threshold of needs.

Thomas Delatte

Rédaction

Code

Louis Beaurepaire

Chef de Projet

Mise en Page

Concept

Code

Adélie Prat

Mise en Page

Rédaction

Relecture

Virgile Ghazi

Relecture

Concept

Code