Université Toulouse III - Paul Sabatier
Bâtiment U3
118 routes de Narbonne
31 062 Toulouse cedex 9

INRA Toulouse Midi-Pyrénées
Unité de MIAT – code : 0875
24 chemin de Borde Rouge -Auzeville
CS 52627
31326 Castanet-Tolosan CEDEX

Rapport de stage

Conception et développement informatique d'une interface de couplage sous **R**

Maeva GARCIA

Stage de fin de licence 3 statistique et informatique décisionnelle
Du 1er avril au 30 juin 2013
Soutenance le 1er juillet 2013

Maîtres de stage : Ronan Trépos et Robert Faivre

Tuteur universitaire : Christelle Chaudet

Edité en Juin 2013

Remerciements

Je remercie dans un premier temps Ronan Trépos et Robert Faivre, mes encadrants de stage pour leur soutien et leur aide durant les trois mois de travail. Ils m'ont permis de réaliser ce stage et m'ont consacré du temps afin de répondre à mes questions. Je remercie aussi Juihui Wang, pour son aide et les solutions qu'il a su apporter aux difficultés rencontrées. Enfin, je remercie Hervé Monod pour sa participation au projet.

Je souhaite aussi remercier Régis Sabbatin, directeur de l'unité au sein de laquelle j'ai effectué mon stage, de m'avoir accueilli pour ces trois mois. Je remercie de même Fabienne Ayrignac, ma responsable administrative pour son accueil le premier jour et m'avoir encadré tout au long du stage.

Je remercie la formation SID, de l'université Paul Sabatier de Toulouse pour leurs enseignements sans lesquels réaliser ce stage n'aurait pas été possible. Je souhaite plus précisément remercier Christelle Chaudet pour son encadrement, et sa visite lors du stage.

Pour finir, je remercie les autres stagiaires et les doctorants de leurs sympathies tout au long du stage, plus particulièrement mon amie Leïla pour son soutien constant.

Résumé

Dans le cadre de ma 3ème année de licence SID, Statistique et Informatique Décisionnelle de Toulouse, j'ai effectué mon stage de fin d'étude à l'INRA (Institut National de la Recherche Agronomique) au département *MIA* (Mathématiques et Informatique Appliquées) dans l'unité *MIAT* (Mathématiques et Informatique Appliquées Toulouse). J'ai travaillé au sein de la plate-forme RECORD qui est une plate-forme de modélisation et de simulation pour l'analyse et la conception de systèmes de culture innovants. Elle s'appuie sur un logiciel codé en *C++* nommé *VLE* (Virtual Environment Laboratory) qui est un environnement de multi-modélisation et de simulation.

Le réseau MEXICO est un réseau de scientifiques animé par plusieurs chercheurs du département *MIA* de l'INRA, qui propose un paquet **R** pour l'exploration de modèle en se focalisant sur l'interopérabilité avec des plates-formes de modélisation. Ce paquet propose une homogénéisation de différentes méthodes d'analyse et a vocation à s'étendre à d'autres problématiques. Durant ce stage nous avons ciblé les méthodes d'analyse de sensibilité. Ce type d'analyse permet au modélisateur d'identifier les paramètres et les variables d'entrées qui ont une forte influence sur les sorties et inversement d'identifier ceux qui ont une influence moindre.

La plate-forme de simulation souhaitait proposer à ses différents utilisateurs des outils pour l'évaluation statistique et l'exploration numérique de leurs modèles. L'objectif était donc de réaliser le couplage entre le paquet d'exploration de modèles et le paquet *rvle*, qui assure la communication entre **R** et *VLE*. Le but étant d'utiliser les fonctionnalités du paquet d'exploration de modèle dans le paquet de simulation. J'ai ainsi réalisé le couplage en développant un script **R** permettant d'analyser les modèles avec les commandes disponibles dans le paquet d'exploration.

Le résultat obtenu est l'interface informatique entre les deux paquets. Cette interface informatique permet aux utilisateurs de réaliser l'exploration de modèle en utilisant les fonctionnalités du paquet d'exploration.

Abstract

As part of my 3rd year license SID (statistics decisional and computer science) of Toulouse, I made my final internship at INRA (French National Institute for Agricultural Research) into MIA department (Mathematics and Computer Science Applied) in MIAT unit (Mathematics and Computer Science Applied Toulouse). I worked in the platform RECORD which is a platform for modeling and dedicated to the study of agro-ecosystems. It relies on software coded in C++ named VLE (Virtual Environment Laboratory) which is an environment multi-modeling and simulation.

The network MEXICO is a network animated by several scientific researchers from MIA Department of INRA, which offers a **R** package model exploration focusing on interoperability with modeling platforms. This package provides a homogenization of different methods of sensitivity analysis and is intended to extend to other issues. During this internship we focused on methods of sensitivity analysis. This type of analysis allows the modelisator to identify the inputs parameters that have a major influence on the outputs, and conversely to identify those who have a lower influence.

The platform simulations wanted to propose to its users different tools for the package statistical analysis and numerical exploration of their models. The objective was to realize the coupling between the package of model exploration and *rvle* which provides communication between **R** and *VLE*. I realized the coupling by developing a **R** script that analyzes models with the available commands in the exploration package.

The obtained result is the computer interface between the two packages. This computer interface allows users to perform the model exploration using the features of package of exploration.

Sommaire

Introduction

Mon stage de fin d'étude de troisième année de licence Statistique et Informatique Décisionnelle (SID) de l'université Paul Sabatier, à Toulouse s'est déroulé au sein de l'Institut National de la Recherche Agronomique de Toulouse, sur le site d'Auzeville, dans l'unité de Mathématiques et Informatique Appliquées de Toulouse (MIAT). Il s'est tenu sur trois mois du 2 avril au 28 juin 2013.

L'objectif de ce stage était de concevoir et de développer une interface de couplage entre l'environnement générique développé par le réseau MEXICO et le logiciel de modélisation et de simulation VLE. Il convenait de développer le protocole de couplage entre ces deux outils et de le mettre en œuvre dans le langage **R**. Enfin la dernière partie du travail avait pour mission de réaliser une interface graphique contenant le couplage.

Dans un premier temps, je présenterai l'INRA, ses missions, ses unités ou encore ses développements. Dans un deuxième temps, nous présenterons les techniques utilisées aux cours de ce stage : les outils utilisés et les logiciels déjà présents. Pour poursuivre, nous approfondirons le travail réalisé et les problèmes rencontrés. Enfin avant de conclure, nous aborderons les méthodologies mises en place durant le stage.

Partie I : Présentation de l'entreprise

Dans cette première partie, nous allons présenter l'entreprise qui m'a accueilli pour ce stage : l'Institut National de la Recherche Agronomique.

I – L'INRA

1. Historique

Mon stage s'est déroulé au sein de l'Institut National de la Recherche Agronomique qui fût fondé en 1946 dans le contexte de la reconstruction nationale d'après-guerre et du projet de modernisation de l'agriculture française. L'institut accompagne les changements du monde agricole, des filières alimentaires et des territoires dans l'objectif de répondre aux attentes exprimées par la société. Cet institut se trouve sous la double tutelle du ministère de la Recherche et du ministère chargé de l'agriculture. Afin de répondre aux défis scientifiques et sociétaux qui ont évolués et qui ont une dimension mondiale, l'INRA a renouvelé ses approches. Les recherches de l'institut concernent les questions en rapport à l'agriculture, à l'alimentation et à la sécurité des aliments, à l'environnement et à la gestion des territoires avec une perspective de développement durable.

L'institut possède actuellement un dispositif de recherche décentralisé, et mutualisé comptant près de 8500 agents titulaires (chercheurs, ingénieurs, doctorants et administratif). Ces effectifs sont répartis en 17 centres régionaux et 13 départements scientifiques. L'INRA occupe le deuxième rang mondial et le premier en Europe pour le nombre de publications en science agricoles et en sciences de la plante et de l'animal.

2. Les objectifs

Les principaux objectifs de l'INRA sont les suivants :

- produire et diffuser des connaissances scientifiques dans de nombreux domaines (exemples : les sciences de la vie, les sciences de l'environnement, les mathématiques et l'informatique appliquées…)
- concevoir des innovations et des savoir-faire pour la société qui permettent le développement d'entreprises agricoles, industrielles ou de services. L'institut

permet de favoriser l'emploi, en partageant ses découvertes au plus grand nombre.

- éclairer, par son expertise, les décisions des acteurs publics et privés.
- développer la culture scientifique et technique, participer au débat science/société.
- former à la recherche et par la recherche, grâce notamment à l'accueil de doctorants et à des partenariats établis avec de nombreux établissements de l'enseignement supérieur.

II – Le centre Toulouse Midi-Pyrénées

Dans cette deuxième partie, nous allons présenter le centre de recherche de Midi-Pyrénées.

1. Historique

Le centre de Toulouse se situe à Auzeville, il est l'un des 17 centres INRA réparties sur l'ensemble de la métropole. Il a été crée en 1970, et compte aujourd'hui plus de 850 chercheurs, ingénieurs et techniciens dont 600 titulaires. Il représente environ 10 % des publications et près de 12 % des brevets de l'INRA. Le site est dirigé par Michèle Marin, présidente du centre chargée d'élaborer la politique d'action régionale de l'institut, en cohérence avec la stratégie nationale de l'INRA.

Les recherches de l'institut sont conduites au sein de douze départements scientifiques. Chaque département est défini par le croisement de quelques disciplines scientifiques et thématiques majeures. Le site d'Auzeville a un dispositif de 15 unités de recherches dont 12 en partenariats avec des universités et des établissements d'enseignements supérieurs et de recherches régionaux.

2. Axes scientifiques

Grâce à une large diversité des compétences des sciences du vivant aux sciences du numérique et aux sciences économiques et sociales, les équipes du centre INRA Toulouse Midi-Pyrénées privilégient des recherches pour faire face aux grands défis du vingt-et-unième siècle comme par exemple des systèmes de productions agricoles (végétaux, animaux) et forestiers plus durables adaptés au changement climatique, ou encore une alimentation attentive aux questions de santé.

Ces recherches se déclinent en sept axes scientifiques majeurs :

- la biologie intégrative[1] des interactions plantes-environnements ;
- la génétique et la biologie animale intégrative ;
- la nutrition et prévention ;
- les biotechnologies industrielles ;
- les méthodes et plates-formes pour la biologie intégrative animale, végétale et microbienne ;
- l'agroécologie des territoires agricoles et forestiers ;
- l'économie de l'environnement et des marchés.

Fort d'une large diversité de compétence, le centre de Midi-Pyrénées peut prétendre jouer un rôle majeur en réponse aux enjeux du vingt-et-unième siècle.

III – Département MIA

Il existe plusieurs départements de recherche au sein de l'INRA, mon unité était rattachée au département Mathématiques et Informatique Appliquées.

Le département Mathématiques et Informatique Appliquées a pour principal objectif de mettre au point des méthodes et des outils dans le domaine des mathématiques et de l'informatique, en particulier pour des applications dans le domaine des sciences du vivant et de l'environnement. Il rassemble ainsi des compétences de recherche en bioinformatique, en biologie des systèmes en éco-informatique et aussi dans la représentation et l'analyse de systèmes complexes.

Le département MIA est présent sur 7 centres INRA et compte environ 150 agents.

IV – Unité MIAT

Les différents centres de recherches se décomposent en unité de recherches, et toutes les unités sont rattachées à un département. Mon stage a eu lieu dans l'unité Mathématique et Informatique Appliquée Toulouse.

[1] **La Biologie intégrative** a pour but d'étudier le fonctionnement d'un organisme vivant dans sa complexité. Il s'agit donc de décrire, comprendre, et prédire les êtres vivants dans son ensemble.

1. Objectifs

Les travaux de l'unité ont pour mission de mettre à la disposition de l'INRA des méthodes et des compétences à jour en mathématiques et en informatique appliquées, en particulier dans le cadre de collaborations avec les autres départements dans des projets et des programmes cohérents avec les axes stratégiques de l'institut. Les compétences disciplinaires présentes au sein de l'unité, couvrent un large spectre en statistique, probabilité, algorithmique, intelligence artificielle et science de la décision.

2. L'organisation

L'unité est dirigée par Régis Sabbadin, et composée de six équipes[2]. Parmi les six équipes nous retrouvons :

- deux équipes de recherche qui sont SaAb (Statistique et Algorithme pour la Biologie) et MAD (Modélisation des Agro-écosystèmes et Décision). Elles correspondent respectivement aux domaines suivants : la bioinformatique et la modélisation de systèmes complexes ;
- trois plates-formes[3] : deux plates-formes axées sur la bioinformatique et une plate-forme de modélisation nommée RECORD ;
- une équipe administrative.

Ces recherches s'accompagnent également d'une activité de production de logiciels pour leur valorisation et d'une activité de formation pour leur diffusion. Au sein de l'unité sont développés des logiciels qui peuvent être ciblés vers les chercheurs et la communauté scientifique de référence ou vers les biologistes et les agronomes. Nous allons à présent nous intéresser plus précisément à la plate-forme RECORD car il s'agit de la plate-forme au sein de laquelle j'ai effectué mon stage.

[2] Voir **annexe 2 page 50** : organigramme de l'unité MIAT

[3] **Une Plate-forme** est en informatique une base de travail à partir de laquelle nous pouvons écrire, lire, utiliser, développer un ensemble de logiciels.

V– La plate-forme RECORD

1. Présentation

La plate-forme RECORD[4] de modélisation et de simulation a pour objectif d'aider à la conception et à l'évaluation des systèmes de production. Pour les agronomes, les expérimentations aux champs sont indispensables mais très coûteuses, longues à mettre en place et limitées en matière de généralisation des résultats. C'est pourquoi l'expérimentation virtuelle assistée par modèle (comme le propose la plate-forme) a pris un grand essor dans la profession d'agronome. Cela leur permet de concevoir des systèmes de cultures répondant aux enjeux actuels de l'agriculture comme par exemple le réchauffement climatique ou encore les différents problèmes environnementaux.

Afin de faciliter le travail des chercheurs, la plate-forme fut créée en 2006. Son objectif est de répondre aux besoins des chercheurs dans le domaine de la modélisation et de la simulation de système de culture, en fournissant des outils permettant l'utilisation de modèles hétérogènes tels que les modèles de plantes, les modèles de sol, les modèles de bio-agresseurs, ou encore les modèles de décision.

2. Logiciels développés

Aujourd'hui la plate-forme est opérationnelle et s'articule autour de trois axes clés :

- un logiciel de modélisation et de simulation appelé *VLE* (Virtual Laboratory Environment). Il s'agit d'un environnement générique de modélisation et simulation informatique ;
- des fonctionnalités permettant de se lier dynamiquement à des outils d'intérêt comme le logiciel statistique R ou Python ;
- une bibliothèque de modèles partagée sur un dépôt web ainsi qu'une documentation de tous ces éléments.

3. Fonctionnement de la plate-forme

Le fonctionnement de RECORD est organisé autour de 3 entités : l'équipe plate-forme RECORD (basée dans l'unité MIAT à Auzeville) qui assure la

[4] **RÉnovation et COorDination** de la modélisation de cultures pour la gestion des agros-écosystèmes (RECORD).

maintenance, l'évolutivité de la plate-forme, la formation et le support des utilisateurs, le « Réseau des utilisateurs » de RECORD chargé de l'animation de la communauté et le « Comité stratégique » chargé de l'orientation stratégique de la plate-forme.

RECORD permet de créer de nouveaux simulateurs à partir de modèles déjà existants. Il offre aussi la possibilité de lancer des simulations après avoir spécifié le scénario de simulation (les valeurs d'entrée, éventuellement un plan d'expérience, la durée de simulation, les variables dont on veut observer la dynamique...). Il est également possible de lancer des simulations depuis un programme écrit en R ou depuis une interface web.

Partie II : Technologies

Dans cette deuxième partie, nous présenterons le contexte du stage, et les technologies utilisées au cours de ce stage.

I – Le logiciel **R**

1. Présentation générale

Le logiciel **R** est un logiciel de statistique crée par Ross Ihaka et Robert Gentleman. C'est à la fois un langage informatique et un environnement de travail. Les commandes sont exécutées grâce à des instructions codées dans un langage relativement simple, les résultats sont affichés sous forme de texte et les graphiques sont directement visualisés dans une fenêtre qui leur est propre.

Les avantages de ce logiciel sont nombreux notamment la gratuité et l'utilisation sur plusieurs plates-formes comme Unix, Windows ou encore Macintosh. Toute personne peut contribuer à son amélioration, en y intégrant de nouvelles fonctionnalités ou méthodes d'analyse non implémentées. C'est un outil puissant et complet, particulièrement adapté pour la mise en œuvre informatique des méthodes statistiques.
Il peut paraître plus difficile d'accès car il n'est pas conçu pour être utilisé à l'aide de "clics" de souris dans des menus. L'avantage en est toutefois double :

- l'approche est pédagogique puisqu'il faut maîtriser les méthodes statistiques pour parvenir à les mettre en œuvre ;
- l'outil est très efficace lorsque l'on domine le langage **R** puisque l'on devient alors capable de créer ses propres outils ce qui permet ainsi d'opérer des analyses sophistiquées sur les données.

2. Notion de paquets

Pour pouvoir compléter et augmenter les fonctionnalités du logiciel **R**, des paquets également appelés « packages » ou bibliothèques sont développés. Ils sont généralement dévolus à des méthodes particulières ou à un domaine d'application. Il est possible de construire un paquet pour un usage personnel, ou pour être diffusé.

II – Les paquets utilisés en stage

Lors du stage, j'ai travaillé sur trois paquets **R** différents. Le premier est un paquet de simulation de modèles, le second est un paquet d'exploration de modèles et le dernier paquet permet de créer des interfaces graphiques sous **R**.

1. Le paquet *rvle*, pour la simulation de modèles

RECORD se base sur un projet nommé *VLE* (Virtual Laboratory Environment) existe et a pour objectif de permettre la simulation et la modélisation de systèmes complexes dynamiques. Il a été développé pour proposer un ensemble d'outils et de bibliothèques pour modéliser et simuler des modèles hétérogènes spécifiés dans différents formalismes. Afin de bien comprendre le projet, nous avons besoin d'expliquer rapidement la structure des deux logiciels *VLE* et *GVLE*. Ces deux logiciels ont été développés dans le langage de programmation orientée objets *C++*.

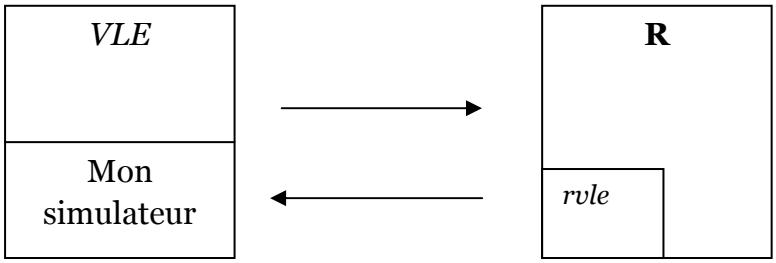
VLE est un environnement générique de multi-modélisation et de simulation de systèmes complexes dynamique informatique. C'est un ensemble d'outils et de bibliothèques qui permet de coupler et de simuler des modèles hétérogènes, c'est-à-dire spécifiés dans des formalismes différents. Cet environnement est conçu sur les bases théoriques et opérationnelles de la modélisation et de la simulation. Les modèles atomiques sont décrits comme un ensemble d'états, un ensemble d'entrées et de sorties et une dynamique. Les modèles atomiques se connectent entre eux via les entrées et les sorties pour former les modèles couplés.

Chaque modèle possède un fichier *vpz* qui est un fichier *xml* permettant de décrire les éléments d'une expérience. Il permet de définir la structure des modèles, les conditions expérimentales, ainsi que les éléments observés et la méthode d'observation.

GVLE est une interface graphique pour *VLE*, permettant de simplifier la création, l'édition et la simulation de modèles. Ce logiciel vise à cacher la configuration du fichier *vpz* et les différentes étapes de configurations et de compilations. Cela permet d'ouvrir la plate-forme à un plus grand nombre d'utilisateurs.

Ces deux logiciels nous permettent donc de visualiser, modéliser et simuler un modèle, mais RECORD a également mis en place un autre outil permettant d'analyser les simulations des modèles depuis **R**. Il s'agit de *rvle*.

L'outil *rvle* est un outil important car il fait partie de la base de travail du stage. Il s'agit d'un paquet sous **R** pour l'utilisation d'un simulateur d'objet de *VLE*. Il permet la communication entre **R** et *VLE*.



Le paquet *rvle* permet depuis une session **R** de :

- lire des fichiers *vpz*,
- de modifier les conditions initiales et les paramètres des simulateurs et les cas échéants les sauvegarder dans le simulateur,
- de modifier la durée de simulation et d'autres paramètres caractéristiques de la simulation,
- lancer des simulations,
- récupérer les résultats de la simulation sous forme d'objet **R** : matrices ou dataframes.

Ce paquet est utile pour traiter les simulations. Il permet à un utilisateur de « manier » les modèles depuis un environnement simple qui est **R**. Pour ce faire différentes commandes sont spécifiques aux paquets, elles seront développés un peu plus tard dans le rapport.

2. Le paquet *mtk*, pour l'exploration de modèles

Le réseau MEXICO (Méthode pour l'EXploration Informatique de modèles COmplexes) est né en 2006, en rassemblant une communauté d'informaticiens modélisateurs et de statisticiens de différents instituts de recherche et universités. La plupart des équipes avaient déjà des échanges informels autour de la problématique

d'exploration numérique des modèles, il s'agissait donc de donner un cadre à ces échanges et d'identifier les questions communes à ce groupe qui pourraient gagner à être mutualisées.

Ce réseau a 3 objectifs distincts : initier les chercheurs aux méthodes d'exploration de modèles notamment lors des interventions dans les écoles chercheurs, contribuer à la réflexion méthodologique notamment à travers un livre qu'ils ont écrits et paru en début d'année portant sur l'analyse de sensibilité et l'exploration de modèle. Et enfin le dernier but est de rendre accessible ces méthodes. Pour ce faire ils ont mis au point le paquet **R** d'exploration de modèle. Cette boîte à outils à pour but :

- offrir une large sélection de méthodes permettant l'exploration numérique de modèles,
- permettre d'intégrer de nouvelles méthodes pour l'exploration numérique de modèle,
- pouvoir dialoguer avec des plates-formes de modélisation,
- fournir un ensemble de fonctionnalités pour utiliser les différentes méthodes exploratrices des modèles. C'est l'objet du paquet « mtk » que nous allons aborder plus en détails.

Il existe plusieurs projets permettant d'explorer le comportement des modèles et d'exploiter plus efficacement les données qu'ils produisent, au sein de la communauté de chercheurs. Il en résulte de nombreux logiciels néanmoins, ils manquent quelquefois d'universalité ou de généricité dans leur exploitation, notamment quand il est question de les intégrer par la suite à des plates-formes de modélisation (comme *VLE*, par exemple). Par ce constat, le paquet **R** « mtk », a été développé. Il est consacré aux méthodes d'exploration de modèles par la simulation.

Ce paquet se veut être universel et standardisé dans les méthodes d'exploration numérique en réponse aux attentes des plates-formes de simulation. Il a trois vocations principales : pouvoir être utilisé facilement à partir du logiciel **R**, pouvoir s'intégrer aisément dans les plates-formes de simulation, et pouvoir être enrichi facilement par d'autres contributeurs par l'ajout de nouvelles méthodes.

Ce paquet est le résultat d'une collaboration entre des statisticiens spécialisés dans les méthodes d'exploration numérique et des informaticiens spécialisés de la programmation orientée-objet. Il a donc une conception et une architecture d'ensemble originales pour un paquet **R**, qui a pour objet de concilier l'interactivité d'un langage de script comme **R** et l'efficacité de programmation orientée-objet comme Java. Son architecture repose sur une représentation homogène des facteurs d'entrée, caractérisée par des lois et sur une décomposition de l'expérimentation numérique en étapes. Cette architecture sera développée plus en détails dans la suite du rapport car elle fait partie intégrante des éléments de recherche du stage.

3. Le paquet « shiny », création interface en R

Le paquet "shiny" est un nouveau paquet de **R** disponible pour tous les utilisateurs du logiciel permettant de construire des applications web interactives avec **R**. Il permet de construire des interfaces Web avec seulement des lignes de codes depuis **R** sans l'utilisation de JavaScript. C'est une manière simple et efficace de créer des applications interactives où les sorties se mettent à jour au fur et à mesure que les entrées sont modifiées sans nécessiter un rechargement du navigateur. L'avantage de ce paquet est qu'il n'est pas nécessaire de connaître de langage comme HTML ou Java, une simple expérience avec **R** est requise. Il combine la puissance statistique de **R** et la programmation web. Il permet de laisser aux utilisateurs le choix des paramètres d'entrées à l'aide des commandes comme le curseur, les menus déroulants ou encore les champs de textes. Il est également possible d'intégrer facilement toutes sortes de sorties comme des tableaux, des graphiques, ou encore des résumés.

De manière plus générale, "shiny" représente une alternative pour créer des applications Web et est accessible depuis l'environnement de **R**. Il est utilisé pour créer l'interface ou figurant le couplage réalisé lors du stage.

III – Liens entre les différents paquets

Le logiciel *VLE*, à travers son couplage avec **R** (paquet *rvle*) offre la possibilité de manipuler les modèles, plus précisément de piloter l'exécution des simulations, ou encore de récupérer les sorties directement en **R**. Le réseau MEXICO grâce à son paquet *mtk* permet d'utiliser des méthodes d'exploration de modèle. Ce qu'apporte ce paquet répond aux attentes de la plate-forme RECORD en termes d'exploration de modèles.

L'objectif du stage au vu des fonctionnalités des deux paquets consiste à fournir aux utilisateurs du logiciel *VLE*, l'environnement générique développé par MEXICO. De manière plus générale, il convient de réaliser l'exploration des modèles issus de *VLE* en utilisant les fonctions du paquet *mtk* et ainsi faciliter l'analyse des entrées et des sorties des modèles. Cela consiste à créer une interface en ligne de commande (CLI)

Suite au couplage, il convient de créer une interface graphique (CGU) avec le paquet de création d'interface. Cette interface est basée sur le couplage qui a été réalisé.

Partie III : Réalisation du stage

Dans cette partie, nous allons développer les différentes étapes du stage et les résultats obtenus.

I – Prise de connaissance du sujet : Exploration de modèle.

La première étape du stage consistait à prendre connaissance du sujet en approfondissant les points les plus importants comme l'analyse de sensibilité, l'analyse d'incertitude, les différentes méthodes d'analyses utilisées…

1. Les différentes notions théoriques

Il y a plusieurs notions à préciser dans cette partie. La première concerne la notion de modèle. Nous aborderons ensuite l'analyse d'incertitude et l'analyse de sensibilité ainsi que la notion d'indice de sensibilité. Nous finirons par présenter la planification d'expérience dans ce contexte.

Les modèles

Comme nous l'avons vu dans la présentation de la plate-forme, l'expérimentation sur le terrain est importante pour les agronomes mais très compliquée à mettre en place et très coûteuse. Les agronomes utilisent donc des modèles représentant le fonctionnement des cultures. Ces modèles sont souvent utilisés dans différents domaines et constituent un outil d'aide à la décision. Il existe une grande diversité de modèles mais ils sont souvent composés de quatre éléments : des variables d'entrée, des variables de sorties, des valeurs des paramètres et des équations. Nous utiliserons le terme de facteurs pour définir les variables d'entrées et les facteurs.

Comme ils sont des représentations simplifiées de la réalité, les modèles conduisent à des erreurs de prédiction ou à des décisions erronées. Il est donc important de connaître les principales sources d'incertitude d'un modèle et d'analyser leurs conséquences sur les prédictions et l'aide à la décision. Les méthodes d'analyse d'incertitude et de sensibilité constituent alors des outils utiles pour décrire l'incertitude dans les sorties d'un modèle et hiérarchiser l'importance des différents éléments incertains.

Les analyses d'incertitude et l'analyse de sensibilité

L'analyse d'incertitude permet d'obtenir des informations sur l'incertitude associée aux prédictions du modèle, induite par les incertitudes sur les facteurs d'entrées du modèle. Ces informations sont importantes pour juger de la qualité des prédictions du modèle et pour optimiser les variables décisionnelles.

Dans le cadre du stage nous nous sommes intéressés à l'analyse de sensibilité qui permet d'identifier les paramètres et les variables d'entrée qui ont une forte influence sur les sorties du modèle, et inversement, identifier les paramètres et les variables d'entrée qui ont une influence moindre sur les sorties. Elle permet en particulier de hiérarchiser l'importance des différents paramètres incertains d'un modèle. Les résultats d'une analyse de sensibilité permettent de déterminer les paramètres qu'il est nécessaire de connaître plus précisément.

Les indices de sensibilité sont les résultats d'une analyse de sensibilité. Cet indice mesure l'influence d'un facteur incertain sur une variable de sortie d'un modèle. Une valeur d'indice élevée indique que le facteur a une forte influence sur la sortie et inversement une valeur faible indique une faible influence. Le calcul d'indices de sensibilité pour plusieurs facteurs incertains permet de hiérarchiser l'influence de ses facteurs.

Les plans d'expériences

Les méthodes d'analyses de sensibilité sont basées sur la création d'un échantillonnage dans l'espace des paramètres d'entrée afin d'obtenir des échantillons de sortie du modèle numérique. L'échantillonnage repose sur la création d'un plan d'expérience. Il a pour but de minimiser le nombre d'essais à conduire pour obtenir des résultats fiables qui reflètent la variation réelle du phénomène étudié en fonction de ses diverses caractéristiques. Il existe plusieurs types de plans. Néanmoins afin de s'assurer que les domaines de variation des paramètres d'entrée sont bien parcourus, nous utilisons l'échantillonnage par hypercubes latins, connu sous l'acronyme *LHS* (Latin Hypercube Sampling). Ce type de plan consiste à répartir les points de l'échantillon uniformément sur toute l'étendue du domaine de chaque variable d'entrée, afin de remplir l'espace.

Pour réaliser une analyse de sensibilité plusieurs méthodes existent. Nous allons à présent développer un exemple, la méthode de Morris.

2. La méthode de Morris

La méthode Morris utilise une discrétisation de l'espace des facteurs : seuls des points appartenant à une grille régulière multidimensionnelle sont susceptibles d'être échantillonnées. Une seconde caractéristique est que l'influence de chaque facteur X_k est évaluée en comparant des simulations pour lesquelles seul ce facteur X_k a varié. Il d'agit donc d'une méthode dite « OAT » signifiant en anglais « *One At a Time* » traduit par « un par un ». La combinaison des deux principes décrits ci-dessus fait de cette méthode Morris, une méthode robuste et efficace pour l'analyse de sensibilité.

Echantillonnage

Soit un modèle comprenant *K* facteurs, la gamme de variation de chaque facteur est discrétisée en *Q* niveaux $\{ 0, \frac{1}{Q-1}, \frac{2}{Q-1}, \dots , 1\}$. Le croisement de ces niveaux définit un ensemble de Q^K nœuds noté Ω. L'échantillonnage de la méthode Morris est constitué d'une suite de trajectoires aléatoires ρ_i pour i =1, …, r passant chacune par *K+1* nœuds $(A^{(i,o)},\dots,A^{(i,K)})$ de Ω, de telle sorte que chaque facteur ne varie qu'une seule fois par trajectoire.
Cette méthode nécessite *l* nombre de simulation où *l= K+1 * r*.

Principe

En analysant les valeurs simulées sur les *r* trajectoires, cette méthode permet de classer les entrées en trois catégories :

- Entrées ayant des effets négligeables,
- Entrées ayant des effets linéaires et sans interaction,
- Entrées ayant des effets non linéaires et/ou avec interaction.

Calcul des indices

La méthode repose sur l'étude des variations de la sortie entre deux points successifs sur les trajectoires. Chaque répétition *i* (avec *i*=1…*r*) permet d'évaluer un effet élémentaire, noté $E_j^{(i)}$ par entrée X_j. Par exemple, supposons que le déplacement entre deux points successifs $A^{(i,j)}$ et $A^{(i,\ j+1)}$ de la trajectoire ρ_i soit associé à une

variation $\epsilon\delta$ $(\epsilon = \pm 1)$ du facteur X_k. Cette variation occasionne l'effet élémentaire suivant :

$$E_j^{(i)} = \frac{g\left(x_1^{(i)},\ldots, x_k^{(i)}+\epsilon\delta,\ldots, x_K^{(i)}\right) - g(x_1^{(i)},\ldots, x_k^{(i)},\ldots, x_K^{(i)})}{\delta}$$

Où :

g est le modèle ;

$x_k^{(i)}$(respectivement $x_k^{(i)} + \epsilon\delta$) désigne la coordonnée sur l'axe associé au facteur X_k du point $A^{(i,j)}$ (respectivement $A^{(i,\, j+1)}$).

L'ensemble du plan d'expérience (r répétitions) fournit un r-échantillon des effets pour chaque entrée X_j, dont sont issus les indices de sensibilité. Nous calculons deux paramètres :

- $\mu_j^* = \sum^r{}_{i=1} | E_j^{(i)} |$ la moyenne des valeurs absolues des effets ;
- $\sigma_j = \sqrt{\frac{1}{r-1}\sum_{i=1}^{r}(E_j^{(i)} - \mu_j)^2}$ l'écart type des effets.

où $\mu_j = \sum^r{}_{i=1} E_j^{(i)}$

Ainsi, plus μ_j^* est important, plus l'entrée X_j contribue à la dispersion de la sortie, le paramètre σ_j mesure quant à lui la linéarité du modèle étudié et les interactions entre les paramètres. Si la sortie dépend linéairement de X_j et que X_j n'interagit pas avec d'autres entrées ($k \neq j$), l'effet d'une perturbation élémentaire de X_j est identique quelle que soit sa position dans l'espace des entrées : les r effets élémentaires sont égaux et σ_j est alors égal à 0. Par conséquent, plus σ_j est élevé (par rapport à μ_j^*), moins l'hypothèse de linéarité et de non interaction est pertinente.

3. Processus d'exploration de modèle proposé par *mtk*

Pour réaliser différentes expérimentations numériques, un enchaînement de quatre procédures est nécessaire : nous l'appelons *workflow*. Il correspond à l'ensemble des étapes à mettre en œuvre pour réaliser l'exploration d'un modèle. L'enchaînement des processus est illustré sur la figure suivante :

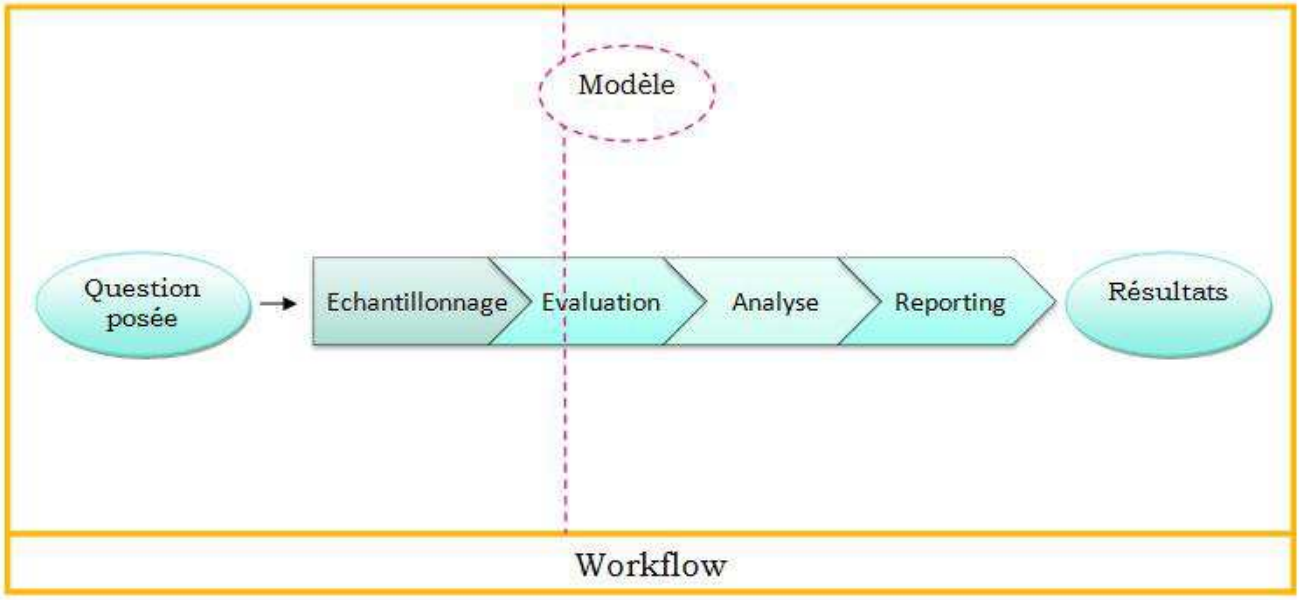


Figure I.1 : processus pour l'exploration d'un modèle

Ce *workflow* est constitué de quatre processus différents pour l'exploration numérique. Le premier processus est **l'échantillonnage,** qui consiste à produire une série de scénarios c'est-à-dire un jeu de valeurs des différents facteurs dont nous souhaitons étudier l'influence. Ce processus construit un plan d'expérience.

Le deuxième processus est **l'évaluation**, qui consiste à exécuter une simulation du modèle pour chaque scénario établi auparavant.

La troisième étape du *workflow* est **l'analyse,** qui consiste à mettre en œuvre la méthode d'analyse souhaitée sur les résultats de l'évaluation. Les données obtenues lors de l'évaluation sont donc analysées par la méthode choisie.

Enfin la quatrième étape est **le reporting,** qui consiste à réaliser une synthèse des résultats de l'analyse effectuée notamment les résumés statistiques ou encore les sorties graphiques.

Ce *workflow* est important car il coordonne les enchaînements des différents processus qui interviennent dans la procédure d'expérimentation numérique. Il a pour rôle d'assurer le bon enchaînement de ces processus. En effet, avant d'appeler un processus à participer à un enchaînement, il vérifie que toutes les conditions

préalables sont remplies et que les données nécessaires à l'enchaînement sont disponibles.

Dans la version du paquet d'exploration que j'ai utilisé, seulement trois types de processus sont pris en compte : l'échantillonnage, l'évaluation et l'analyse. Seules quelques fonctions de reporting sont disponibles.

II – Couplage sur un modèle exemple

La première étape du codage, consistait à réaliser le couplage du *workflow* d'analyse de paquet *mtk* de **R**, sur un modèle exemple issu de la plate-forme. Cela nous a permis de comprendre le fonctionnement des deux paquets ensemble : le paquet de modélisation (*rvle*) et le paquet d'exploration de modèle (*mtk*).

Nous allons, dans un premier temps, définir le modèle exemple choisi puis nous présenterons le couplage sur cet exemple.

1. Présentation du modèle

Le modèle qui a été utilisé tout au long du stage se nomme « 2CV ». Il simule la conduite d'une culture de maïs ainsi que la croissance du maïs et la marge brute économique issue de la vente du grain. Nous utilisons la version du modèle qui repose sur des données météorologiques journalières (pluie, température minimale, température moyenne, température maximale…) qui sont lues dans un fichier. Les simulations nous fournissent quotidiennement un indice du développement du maïs. Et en fin de simulation, les observations fournissent un bilan, en partie économique, par exemple le profit, le rendement ou encore le nombre d'irrigations.

Dans ce modèle, nous prenons en compte cinq variables de décision pour étudier comme variable de sortie le profit, c'est-à-dire la marge issue de la vente du maïs. Les variables de décision concernant l'irrigation du maïs sont les suivantes :

- apportEau, quantité d'eau apportée à chaque irrigation ;
- nbJoursAvantRetour, le nombre minimal de jours qu'il est nécessaire d'attendre entre deux activités d'irrigation ;
- quantiteEauSolPortant, une quantité d'eau de pluie cumulée sur une courte durée au delà de laquelle on considère que le sol n'est plus portant. Dès lors, les activités aux champs sont alors suspendues ;
- quantiteEauPeriodeSeche, une quantité d'eau de pluie cumulée sur une courte durée en dessous de laquelle nous considérons que nous sommes en période sèche et qu'une irrigation est nécessaire ;
- stockEau, la quantité d'eau disponible pour l'irrigation, sur l'ensemble de l'année culturale.

Ce modèle prend donc en paramètre d'entrée les cinq variables décrites ci-dessus afin d'étudier le profit de la culture du maïs.

Après avoir pris en main le modèle, il s'agissait de réaliser le couplage entre le paquet de simulation *rvle* et le paquet d'exploration *mtk* spécifiquement pour ce modèle « 2CV ».

2. Couplage sur ce modèle

Comme nous l'avons stipulé lors de la présentation des différents paquets, le paquet d'exploration de modèle a été mis au point dans l'optique de pouvoir être utilisé au sein de plusieurs plates-formes. Il peut donc être enrichi facilement afin de répondre aux attentes des différents utilisateurs.

Dans le paquet de base, plusieurs modèles sont déjà présents et fonctionnels. Pour pouvoir réaliser l'analyse de sensibilité sur le modèle de conduite du maïs, nous devions l'ajouter au paquet d'exploration de modèle.

Pour ajouter ce modèle à ce paquet, il convient d'écrire deux fonctions distinctes dans un script **R** :

- une fonction « .model » (a)
- une fonction « .simule »(b)

Ces deux fonctions sont requises par le paquet d'exploration de modèle pour réaliser le couplage entre les deux paquets.

a - Fonction « .model »

La fonction « .model » [cf. Figure II.1] permet de récupérer pour un vecteur de paramètre en entrée, une simulation du modèle et surtout la valeur de la variable de sortie : le profit dans notre exemple.

```
cv.model <- function(param){

#simulation du modèle
  cv  =  new("Rvle",  file  =  "2CV-exploration.vpz",  pkg  =
"tp5_4_correction")

setDefault(cv,"condDecAgent.apportEau"=param[1])
setDefault(cv,"condDecAgent.nbJoursAvantRetour"=as.integer(par
am[2]))
setDefault(cv,"condDecAgent.quantiteEauPeriodeSeche"=param[3])
setDefault(cv,"condDecAgent.quantiteEauSolPortant"=param[4])
setDefault(cv,"condDecAgent.stockEau"=param[5])

cvres=results(run(cv))
y<-cvres$vueFinalBilan[1,"model:Bilan.Profit"]
return(y)
}
```

Figure II.1 : code de la fonction « model »

En paramètre d'entrée de cette fonction, nous retrouvons un vecteur nommé ici « param » qui correspond à un scénario de simulation, c'est-à-dire une valeur pour chaque paramètre. Comme notre modèle présente cinq variables de décision, le paramètre « param » est donc un vecteur de dimension 5.

Dans cette fonction, la première étape consiste à déclarer le modèle de conduite de culture du maïs (« 2CV ») comme notre modèle à analyser. Nous utilisons une fonction du paquet de simulation de modèle qui est la fonction *new()*. Cette fonction « new » prend en paramètre le type du modèle, le nom du fichier contenant le modèle et le paquet où est situé ce modèle.
Nous initialisons la variable « cv » comme un objet *rvle* du fichier « 2CV-exploration.vpz » du paquet « tp5_4_correction ». Les deux derniers renseignements sont spécifiques au nom du fichier et au nom du paquet qui contenait le modèle de conduite du maïs sur mon poste. La variable « cv » contient le modèle complet. Nous pouvons voir en exécutant juste « cv »[5] les différentes variables, les valeurs nominales de ces variables et leur type.

La deuxième étape de la fonction « .model » consiste à initialiser les valeurs des variables de décision avec les valeurs du paramètre « param » passé en entrée. Pour cela nous utilisons la fonction *setDefault()* issue du paquet de simulation *rvle*. Cette fonction permet de définir les différents paramètres de simulation. Elle nous permet donc pour chaque variable de décision de lui affecter sa valeur correspondante du vecteur « param ».

La troisième étape de la fonction « .model » est de lancer la simulation du modèle. Nous utilisons la fonction « run » qui permet d'exécuter le modèle présent dans « cv ». Nous utilisons en même temps que la fonction « run » la fonction « results » : elle permet de conserver les résultats, ici la variable « cvres ».

A l'issue de ces trois étapes, nous avons donc chargé un modèle dans « cv », affecté les valeurs passées en paramètre aux variables du modèle, lancé la simulation et conservé les résultats. La dernière étape consiste à récupérer la valeur de sortie. Nous récupérons la valeur du profit dans la variable « cvres ». La variable « cvres »

[5] Voir **annexe 1 page 49** : exemple de contenu de la variable « cv »

est structurée en différentes vues. De manière générale, les sorties de simulation des modèles, sous RECORD, sont structurées en vues ; une vue contenant les informations issues de chaque simulation et une vue contenant le bilan de la simulation. Pour récupérer la valeur qui nous intéresse dans notre cas, nous nous positionnons dans cvres puis sur la vue contenant le bilan en indiquant que nous souhaitons récupérer le profit. C'est ce que fait la commande :

« y<-cvres$vueFinalBilan[1,"model:Bilan.Profit"] ».

Enfin la dernière étape de la fonction « .model » a pour objectif de retourner la valeur de la variable de sortie.

b - Fonction « .simule »

La deuxième fonction nécessaire pour créer le modèle au sein du paquet d'exploration de modèle est la fonction « .simule » [cf. Figure II.2]. Cette fonction permet de simuler le comportement du modèle pour un ensemble de valeurs de paramètres.

```
cv.simule <- function(X,...){
     Y=apply(X,1,cv.model)
     Y=as.data.frame(Y)
     output=list(main=Y,information=NULL)
     return (output)
}
```

Figure II.2 : code de la fonction « simule»

Dans cette fonction, le premier argument X est une matrice à N lignes et K colonnes, avec N le nombre de simulations à effectuer et K le nombre de paramètres étudiés (dans notre exemple, K=5). Chaque ligne de X contient un jeu complet de paramètres. Dans notre exemple, la fonction ne prend aucun autre paramètre, mais il peut y avoir des paramètres en plus comme par exemple des options de décisions pour certains modèles.

La première étape de la fonction consiste à utiliser la fonction « apply() ». Cette fonction permet d'appliquer une fonction à chaque ligne ou colonne d'un tableau de données. Elle prend 3 arguments dans l'ordre suivant : nom de tableau de données, nombre pour dire si la fonction doit s'appliquer aux lignes (1), aux colonnes (2) ou

aux deux (c(1,2)) et enfin le nom de la fonction à appliquer. Dans notre cas, le tableau de données est la matrice *X* passée en paramètre. Nous appliquons la fonction « cv.model » décrite précédemment à chaque ligne de *X* afin de récupérer le profit du maïs correspondant.

La seconde étape consiste à transformer les résultats en un *data-frame*. Un *data-frame* est une structure **R** où chaque colonne peut avoir son type (contrairement à une matrice dont tous les éléments doivent avoir le même type). Enfin, nous écrivons la sortie sous la forme d'une liste qui contient dans un premier temps Y contenant tous les profits obtenus en fonction des différents jeux de données passée dans X et les informations complémentaires. Ici, les informations sont à « NULL » puisqu'il n'y en a pas.

c - Chargement du modèle dans le paquet d'exploration *mtk*

La dernière étape pour intégrer le modèle au sein du paquet d'exploration de modèle est de créer le modèle. Pour piloter le modèle, les deux fonctions décrites dans les deux derniers paragraphes (cv.model et cv.simule) sont regroupées dans un même fichier (2CVModel.R). Le simulateur est intégré au paquet d'exploration *mtk* avec la fonction « mtk.evaluatorAddons ».

Cette fonction prend comme premiers arguments le fichier contenant les fonctions créées et le nom du modèle pour l'appeler depuis le paquet d'exploration. Elle génère le fichier « mtk2CVEvaluator.R ». Cette nouvelle classe est définie automatiquement dans un fichier extérieur qui doit être chargé pour terminer l'adaptation du code du modèle au paquet.
La fonction « mtk.evaluatorAddons » n'est exécutée qu'une seule fois. La commande « source » du fichier « mtk2CVEvaluator.R » est exécutée lors de l'ouverture de chaque session de travail s'il n'est pas sauvegardé sous **R**.

Les lignes de codes sont présentées sur la figure II.3.

```
#Chargement du paquet exploration de modèle
library(mtk)

#création du modèle
mtk.evaluatorAddons(where="~/2CVModel.R", name="2CV", main =
"cv.simule", authors="Maeva", summary=NULL, plot=NULL,
print=NULL)

#chargement du modèle
source("~/mtk2CVEvaluator.R")
```

Figure II.3: code chargement du modèle.

3. Exploration du modèle exemple

Dans les parties précédentes, nous avons donc créé le modèle [cf. figure II.1 et figure II.2] de simulation d'une conduite de maïs puis nous l'avons intégré au paquet d'exploration de modèle *mtk* [cf. figure II.3]. A présent, nous allons réaliser une analyse de sensibilité du modèle de conduite d'une culture de maïs («2CV »), en décrivant les différentes commandes à mettre en œuvre sous **R**.

Le paquet d'exploration de modèle repose sur une architecture à trois composantes[6] : la gestion des facteurs, la gestion du *workflow* et la gestion des accès aux ressources externes.

La première étape à réaliser pour lancer l'analyse est la déclaration des facteurs. Pour rappel le modèle que nous utilisons contient cinq variables de décision. Pour former les différents facteurs, nous utilisons la fonction « **make.mtkFactor** » qui permet de déclarer un facteur sous *mtk*. Cette fonction prend en arguments plusieurs éléments mais il n'est pas obligatoire de tous les renseigner. Seuls deux éléments sont nécessaires pour construire un facteur : « name » le nom du facteur et « distribname » le nom de la distribution. Il existe d'autres paramètres comme « id » qui est le nom complet du facteur, « unit » donnant l'unité des valeurs du facteur, « nominal » donnant la valeur nominale du facteur, « distriPara » concernant les paramètres de la distribution du facteur, « levels » représentant les niveaux des facteurs.

Le code suivant illustre l'utilisation de la fonction définie ci-dessus : il s'agit de la déclaration d'un des cinq facteurs du modèle, le facteur apport d'eau.

[6] Voir annexe **numéro 3 page 51** : architecture du paquet d'exploration « mtk »

```
fact_apportEau <- make.mtkFactor (
                  name="condDecAgent.apportEau",
                  distribName="unif", nominal=30,
                  distribPara=list(min = 5, max = 50))
```

Figure II.4 : déclaration du facteur « apportEau ».

Pour chaque facteur du modèle, il convient d'écrire sa déclaration à l'aide de cette fonction. Une fois la déclaration de tous les facteurs effectuée, nous conservons ceux-ci dans une structure comme suit :

```
cv_fact <- mtkExpFactors(list(fact_apportEau, fact_nbJour,
                        fact_qtJourPS, fact_qtJourSP,
                        fact_StockEau))
```

Figure II.5 : déclaration des différents paramètres.

La fonction « **mtkExpFactors** » prenant une liste contenant tous les paramètres de l'analyse en argument, permet de définir la liste entière des différentes variables de décisions du modèle.

L'étape suivante a pour objectif de définir le processus à mettre en œuvre pour former le *workflow*. Il convient alors de déclarer les trois processus présents dans la version du paquet d'exploration : le générateur de plans, le simulateur et l'analyseur.

```
#Formation du processus de génération des plans
cv.design <- mtkNativeDesigner ("Fast",
                                information=list(n=100))

#Formation du processus de simulation avec le modèle 2CV
cv.evaluate <- mtkNativeEvaluator ("2CV")

# Formation du processus de calcul des indices de sensibilité
cv.analyse <- mtkNativeAnalyser ("Fast",
                                 information=list(nboot=20))
```

Figure II.6 : déclaration des étapes du *Workflow*.

La fonction « **mtkNativeDesigner** » est utilisée pour déclarer la génération des plans d'expérience. Cette fonction prend en paramètre le nom de la méthode utilisée

pour générer les plans et une liste fournissant des informations complémentaires sur la méthode choisie. Cette liste est différente selon la méthode sélectionnée dans le premier argument. Dans l'exemple, il s'agit d'un échantillonnage par Fast.

La fonction « **mtkNativeEvaluator** » permet de mettre en place le processus d'évaluation du modèle et prend en paramètre le nom du modèle que l'on souhaite simuler.

La fonction « **mtkNativeAnalyser** » procède au calcul des indices sensibilité. Elle prend en paramètres la méthode choisie pour l'analyse et une liste d'informations supplémentaires sur la méthode.

Des lors que ces trois processus sont déclarés, il convient d'utiliser la fonction « **mtkExpWorkflow** » pour former le *workflow*. Nous passons en argument de cette fonction deux éléments : la liste contenant tous les facteurs dont nous souhaitons analyser les effets (expFactors) et les différents processus déclarés (processesVector). Ensuite, nous exécutons le *workflow* à l'aide de la fonction « run ». Puis nous récupérons le résumé statistique et les sorties graphiques du modèle.

```
#Formation du workflow.
cv.work <- mtkExpWorkflow(expFactors=cv_fact,
                          processesVector=c(design=cv.design,
                          evaluate=cv.evaluate,
                          analyze=cv.analyse))
#Exécution du workflow.
mtk::run(cv.work)

#résumé statistique et graphique.
summary(cv.work)
plot(cv.work)
```

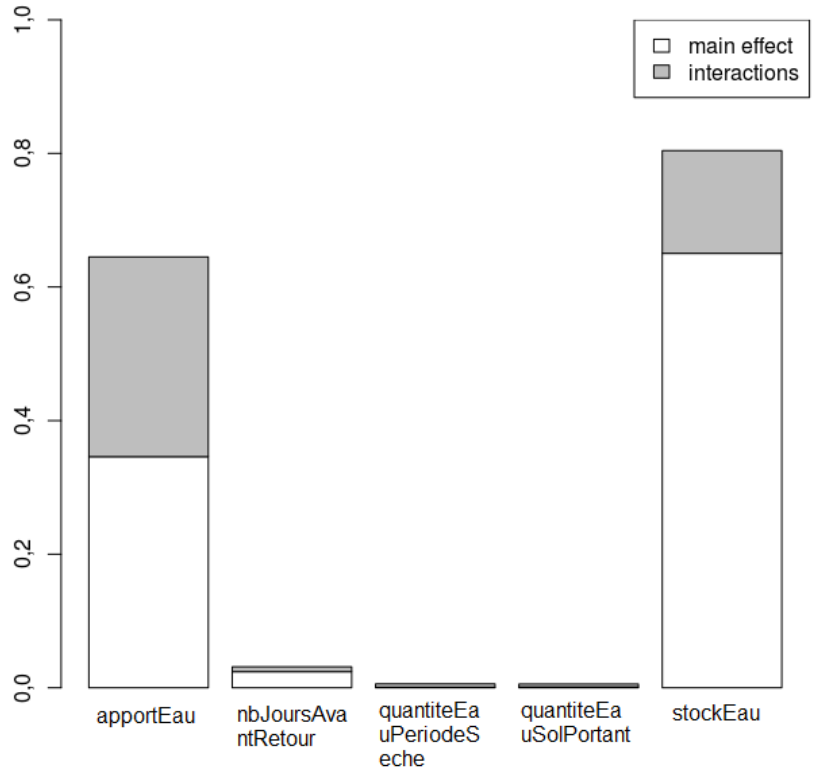
Figure II.7 : étapes sur le *workflow*.

En exécutant les deux dernières lignes de la figure II.7, nous récupérons plusieurs informations intéressantes pour l'analyse de sensibilité comme les indices de sensibilité ou encore le graphique des représentations de l'influence.

	Indices principaux	Indices totaux
apportEau	**0.3597876315**	**0.653138722**
nbJoursAvantRetour	0.0288302419	0.033718844
quantiteEauPeriodeSeche	0.0003402013	0.03487332
quantiteEauSolPortant	0.0002199106	0.004014823
stockEau	**0.6502731658**	**0.822371626**

Tableau II.1 : tableau résumé des indices de sensibilité

Ce tableau nous informe sur les indices de sensibilité principaux et sur les indices de sensibilités totaux. Les indices de sensibilité totaux prennent en compte l'effet principal du facteur et l'effet des interactions avec les autres facteurs. Sur ce tableau, nous pouvons dire que deux facteurs ont le plus d'influence sur notre sortie, ce sont l'apport d'Eau et le stock d'Eau. Nous pouvons retrouver ces résultats graphiquement (graphique II.1).



Graphique II.1 : représentation graphiques de l'influence des facteurs.

Pour conclure, sur notre modèle exemple, cette analyse nous permet de dire que deux facteurs ont une forte influence sur le profit du maïs et inversement trois facteurs ont une influence moindre. Nous pouvons ainsi en déduire que l'apport eau

et le stock d'eau jouent un rôle important sur le profit du maïs. De plus pour la variable apport d'eau nous pouvons constater que ses interactions avec les autres variables ont un effet important sur la sortie. Par contre, les trois variables de décisions (nbJourAvantRetour, quantiteEauPeriodeSeche, quantiteEauSolPortant) n'ont que très peu d'effet sur le profit simulé.

4. Bilan du couplage-exemple

Cette première étape nous a permis d'une part de comprendre le couplage à réaliser, d'autre part nous avons pu constater quelques difficultés. Nous avons rencontré deux principales difficultés dans le couplage de *mtk* et *rvle*.

La première concerne l'appel de certaines fonctions définies dans les deux paquets et donc en conflits de noms. Ce problème a été assez simple à résoudre : il a suffit de préfixer la fonction qui posait problème avec le nom du paquet auquel elle appartient. Par exemple pour exécuter la fonction « run », qui est fournie par les deux paquets, au sein du paquet d'exploration de modèle nous utilisons la commande `mtk::run(workflow)`.

La seconde difficulté à laquelle nous avons du faire face concerne le type des variables que nous souhaitons analyser. En effet, les modèles peuvent prendre en argument différents types de facteurs : des entiers, des réels, des caractères… Le paquet d'exploration *mtk* initialement développé ne permettait de considérer que des facteurs à valeurs réelles. Pour résoudre ce souci, nous avons dû faire appel au concepteur du paquet d'exploration de modèle et nous avons convenu ensemble d'une solution. Nous avons choisi de rajouter un argument lors de la déclaration des facteurs dans la fonction « **make.mtkFactors** » : il est maintenant possible de spécifier le type de la variable. Cela nous permet d'utiliser par la suite deux fonctions « **getType** » et « **setType** » qui permettent de récupérer le type et de modifier le type.

Nous allons à présent expliquer la généralisation de ce couplage.

III – Couplage générique

Dans cette partie, nous allons expliquer la généralisation du couplage entre les deux paquets. L'objectif principal du couplage était de créer une nouvelle version de simulateur sous le paquet d'exploration *mtk*, permettant d'identifier chaque modèle de la plate-forme de modélisation RECORD.

1. Démarche de développement

Dans cette partie, nous ne développerons pas le code qui a été crée, celui-ci sera mis en *annexe numéro 4 page 52* de ce rapport. Nous expliquerons les principales démarches de développement et les différences par rapport au couplage sur l'exemple.

Le simulateur que nous souhaitions créer devait être unique pour tous les modèles de RECORD mais configurable pour l'identification du modèle. De la même manière que pour le couplage sur l'exemple, nous avons du écrire une fonction « .simule » [cf. figure II.2] ayant le même objectif c'est-à-dire l'évaluation du modèle pour un ensemble de valeurs des paramètres. Lors de cette première étape nous n'avions renseigné aucun élément dans l'argument information de la fonction. Dans la généralisation de la méthode, nous avons choisi d'utiliser cet argument pour spécifier l'identification du modèle. Nous précisons différents arguments : le paquet *VLE* auquel appartient le modèle, le nom du modèle dans ce paquet *VLE*, la vue, un indice et le nom de la variables de sortie. Nous passons également dans cet argument la liste contenant tous les facteurs [cf. figure II.5]. Elle va nous être utile pour spécifier le type des variables avant la simulation, si nous ne spécifions pas le type, la simulation du modèle au sein du paquet de simulation peut échouer.

Lors du couplage sur l'exemple, nous avons créé une fonction « .model » [cf. figure II.1] qui avait pour but de simuler le modèle en fonction des paramètres passés en argument. Dans le couplage générique, la fonction « .model » n'existe pas. Ce choix est justifié car nous souhaitions utiliser la simulation des plans d'expérience et la parallélisation fourni par le paquet de simulation de modèle *rvle*. Nous ne souhaitions pas utiliser la fonction *apply()* [cf. figure II.2].

C'est au sein de cette fonction [cf. annexe 4 page 52] que nous retrouvons le couplage entre les deux paquets. Elle nous permet d'étendre les fonctionnalités de la bibliothèque d'exploration de modèle *mtk* au paquet de simulation *rvle*.

Dans la partie qui suit, nous allons présenter les différents changements qu'impliquent le couplage générique pour effectuer une analyse de sensibilité.

2. Modification lors de l'analyse

La première modification concerne la déclaration des facteurs, il nous faut rajouter le type lors de l'appel de la fonction « make.mtkFactors » [cf. annexe 5 page 53]. Il y a une seule modification apportée pour cette fonction, nous rajoutons le type du facteur.

Lors du chargement du modèle, nous prenons en compte le fichier **R** contenant notre fonction « RvleMtk.simule » [cf. annexe 6 page 54] et nous appelons ce simulateur « rvle », au lieu de « 2CV » lors du couplage exemple [cf. figure II.3] dans la fonction « mtk.evaluatorAddons » par la création du « mtkrvleEvaluator.R ». L'utilisateur de RECORD n'a ensuite qu'à charger la nouvelle classe de simulateurs en l'intégrant dans la session par la commande « source » [cf. annexe 6 page 54]. Il est envisagé qu'elle soit intégrée au paquet d'exploration *mtk* par son concepteur dans sa prochaine version (release).

La dernière modification concerne la déclaration du processus de simulation du *workflow* [cf. annexe 7 page 53]. C'est l'appel de cette fonction qui est très important. Elle nous permet de définir le simulateur *rvle*. Nous retrouvons au sein de cette déclaration, les éléments d'identification du modèle

Il n'y a pas d'autres changements pour réaliser l'analyse de sensibilité. Le reste du code reste inchangé pour lancer l'analyse.

3. Bilan de couplage générique

La structure du paquet d'exploration de modèle *mtk* permettait de réaliser le couplage avec le paquet de simulation *rvle*. D'une part le paquet d'exploration est flexible et permet d'intégrer de nouvelles méthodes, d'autre part cette extension est possible pour la plate-forme RECORD. La structure de la fonction comportant le couplage permet d'ajouter les informations nécessaires pour identifier le modèle, les vues et la variable de sortie. L'interface informatique développée, dans cette partie, permet de réaliser l'analyse de sensibilité sur l'ensemble des modèles issus de *VLE*.

IV – Interface graphique

Dans cette partie, nous allons aborder le dernier aspect du stage qui est l'interface graphique. Cette interface complète l'interface fonctionnelle réalisée auparavant. Elle a été réalisée avec le paquet de création d'interface **R**, *shiny*. Nous expliquerons dans un premier temps, les objectifs de l'interface puis nous aborderons la démarche de développement.

1. Objectif de l'interface

La première démarche du stage a consisté à réaliser une interface informatique entre deux environnements. Concernant une question d'utilisation, une interface graphique est plus adaptée à des utilisateurs. Dans ce cadre, le projet de créer une interface graphique est apparu. Elle permettra, une fois terminée, de prendre en compte le couplage qui a été réalisé.

Au sein de la plate-forme RECORD, un projet contenant une interface graphique pour simuler des modèles existait déjà. Elle avait pour ambition de fournir un environnement plus adapté à des utilisateurs non spécialistes du langage **R**. L'objectif final a pour but de fournir une interface permettant en plus de lancer de simple simulation, d'explorer les modèles. Cependant, elle n'est pas encore terminée.

2. Démarche de développement

La conception d'une interface avec ce paquet **R** repose sur deux fichiers : un fichier pour l'aspect graphique de l'interface (ui.R) et un fichier pour l'aspect contenu de l'interface (server.R).

a) Fichier « ui.R » contenant la description de l'interface

L'interface que nous souhaitions adopter était constituée d'un panneau latéral gauche où les options sont définies par les utilisateurs. Ce panneau permet de préciser l'expérience que souhaite réaliser l'utilisateur : l'analyse de sensibilité dans notre cas. Par la suite nous précisions les informations pour récupérer le modèle que nous souhaitons analyser. Et les dernières informations concernent les facteurs à prendre en compte dans l'analyse.

Une maquette se trouve en *annexe numéro 8 page 54*. Pour décrire un minimum ce panneau situé à gauche, il se compose comme un formulaire où plusieurs boutons

apparaissent. En premier, trois listes déroulantes sont affichées : la première (« *Experiment* ») pour sélectionner l'analyse que nous souhaitons réaliser (dans notre cas « *Sensitivity Analysis* »), la seconde (« *Packages* ») pour sélectionner le paquet *VLE* dans lequel se trouve le modèle et la troisième (« *Model* ») pour sélectionner le modèle.

La deuxième partie des boutons concerne les facteurs à prendre en compte : nous avons utilisé des cases à cocher cliquables par l'utilisateur pour chaque facteur du modèle afin de différencier ceux concernés par l'analyse à réaliser des autres. Si la case est cochée, le facteur sera pris en compte pour l'analyse de sensibilité. Si ce facteur est pris en compte, il nous faut renseigner plusieurs informations sur lui (sa valeur nominale, le type de la variable, la distribution et les paramètres de cette distribution). Nous affichons ainsi pour chaque facteur un bouton d'option (bouton radio) pour le type de la variable et la distribution. Nous affichons également un champ de texte pour la valeur nominale et les paramètres de distribution.
Les trois derniers boutons sont des champs de textes permettant de renseigner la variable de sortie du modèle, la vue et l'indice de vue.

L'interface comporte également un panneau de droite qui est la partie principale, où les résultats sont affichés. Ce panneau nous permet de faire figurer les sorties que nous souhaitons afficher. Pour l'analyse de sensibilité, nous retrouvons sur ce panneau les différentes sorties que nous a fourni **R** suite à l'exécution de l'analyse.

b) Fichier « server.R » contenant le contenu de l'interface

Le script du fichier « server.R » réagit aux changements de l'entrée collectée dans le fichier « ui.R » et met à jour les sorties au fur et à mesure des modifications. C'est ce fichier qui permet de fournir une réactivité aux actions de l'utilisateur. Les entrées peuvent être des valeurs numériques, des trames de données ou encore des graphiques…

Ce script est composé de trois fonctions principales :

- une fonction pour récupérer les différents paquets *VLE* contenant les modèles ;
- une fonction pour récupérer les modèles ;

- une fonction pour récupérer les différents facteurs des modèles ;

3. Perspectives

A la fin du stage, l'interface n'est pas terminée il reste encore à faire le lien entre cette interface et le couplage de manière à récupérer les informations sur les différents analyses que nous souhaitons faire. Il faut faire le lien entre ce qui a été codé et les informations saisies dans l'interface.

Partie IV : Méthodologies

Dans cette partie, nous allons expliquer les méthodes utilisées lors du stage, plus précisément l'organisation du travail, ou encore les différentes méthodes employées pour communiquer avec mes responsables.

I – Planning

Ce premier point, a pour objectif de présenter le diagramme de Gantt illustrant le planning prévisionnel et le planning effectif. Il a pour but de montrer les différentes étapes du stage et leur durée.

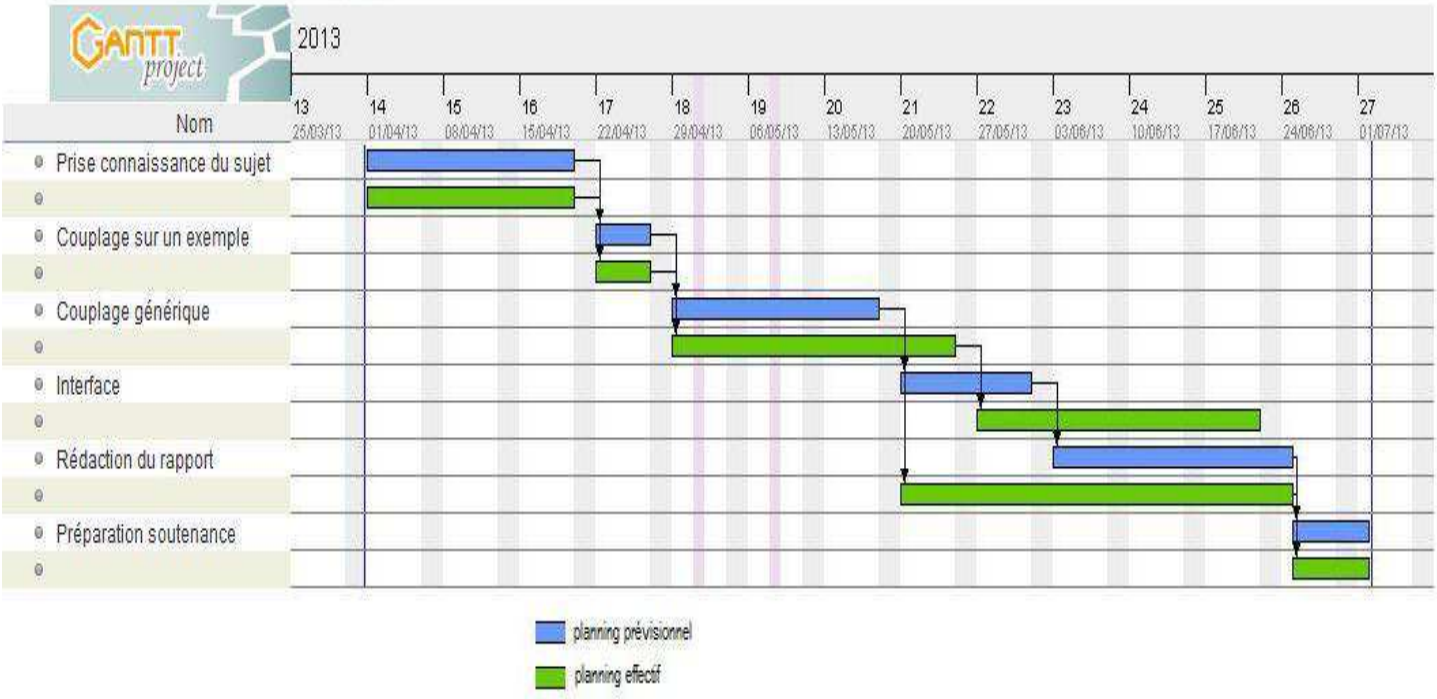


Figure I.1 : diagramme de Gantt

Ce diagramme nous permet de constater qu'il y a eu trois semaines de compréhension du sujet : au cours de ces trois semaines, j'ai beaucoup lu de documentation sur le contexte du stage, les différentes fonctionnalités des réalisations déjà présentes. J'ai notamment dû comprendre les enjeux d'une analyse de sensibilité par exemple. Durant cette période, je me suis notamment « familiariser » avec les deux environnements sur lesquels j'ai travaillé : l'exploration

de modèle et la simulation de modèle. Comme nous le montre le diagramme, le temps d'adaptation et d'appropriation du sujet a été le même que celui prévu, je pense avoir très bien compris rapidement le sujet et m'être habitué rapidement à l'environnement de travail proposé.

Le premier exercice de codage, le codage sur un exemple, comme il a été présenté dans la partie précédente avait été programmé sur une période d'une semaine. Ce délai a été respecté, et le codage générique a pu commencer à la date prévue.

Le couplage généralisé a duré plus longtemps que prévu car il a posé des problèmes auxquels nous avons dû consacrer plus de temps qu'envisagé.

L'interface graphique n'était pas présente sur la proposition de stage initiale. Elle a été rajoutée au bout d'un certain de temps, en vue de mon adaptation. Néanmoins, l'objectif n'était pas de la finaliser mais de l'avancer. Il aurait été ambitieux de penser la finir durant un stage de trois mois.

Enfin la dernière étape du stage, est la rédaction de ce rapport, nous pouvons constater que je l'ai commencé une semaine plus tôt que ce qui était annoncé : cette avance est notamment dû à la journée des stagiaires qui a eu lieu le 13 juin au sein de l'unité, journée durant laquelle j'ai dû présenter mon travail et réaliser un résumé. En préparant cette journée, j'ai donc commencé le regroupement des mes travaux et leur rédaction.

II– Moyens à disposition et mis en œuvre

Nous allons à présent expliquer les moyens mis à la disposition des stagiaires durant les stages, ainsi que les méthodes de communication utilisées.

1. Poste de travail

Afin d'optimiser les conditions du stage, l'INRA et l'unité MIAT met à la disposition des stagiaires un certain nombre d'outils.

Dans un premier temps, un poste de travail personnel sous un système d'exploitation linux, qui est Ubuntu. L'utilisation d'un système linux a été un avantage pour réaliser ce stage. Notamment grâce au terminal, autrement dit la console, qui a permis d'effectuer de manière assez simple des opérations importantes comme la compilation de projet ou encore l'utilisation du GIT.

De plus, assimilée à notre compte INRA, une adresse email a été crée. Elle a permis de communiquer avec les autres développeurs et de recevoir les informations sur la vie de l'unité ou plus généralement sur la vie du centre.

2. Méthode *AGILE*

Pour réaliser ce projet, un cycle de développement s'inspirant des méthodes *AGILE* a été mis en place. Il vise à réduire le cycle de vie du logiciel, donc accélérant son développement, en mettant en œuvre une version minimale. C'est ce que nous avons fait en commençant le couplage sur un exemple. Par la suite, nous avons généralisé le couplage en intégrant des nouvelles fonctionnalités afin d'obtenir une version stable du projet. L'origine de ce type de méthodes est liée à l'instabilité de l'environnement technologique et au fait que le client est souvent dans l'incapacité de définir ses besoins de manières exhaustives dès le début du projet. Le terme « agile » fait référence ainsi, à la capacité d'adaptation aux changements de contexte et aux modifications de spécifications intervenant pendant le processus de développement. Ces méthodes permettent aux clients d'être pilotes du projet et obtenir assez rapidement une première production du développement souhaité.

3. *GIT* et le processus d'intégration

Le développement des différents projets au sein d'une plate-forme comme RECORD s'appuie sur un travail collaboratif. En effet, les travaux de développement sont découpés en tâches, que se partagent les développeurs. Afin de garantir un processus de développement rigoureux et durable, une organisation à plusieurs niveaux d'intégration est en place.

a) Généralités

Le système de gestion de version choisi est *GIT*, il permet de conserver l'historique des modifications apportées dans le développement d'un logiciel. Il est ainsi plus aisé de savoir quel développeur a effectué une modification, quand celle-ci a été effectuée et en quoi elle consistait.

GIT possède la particularité de fonctionner sur un mode décentralisé. Cette caractéristique apporte de nombreux avantages par rapport à d'autres systèmes de gestion de version (comme CVS par exemple), notamment :

- travailler de manière désynchronisée par rapport à d'autres systèmes de gestion de version ;
- travailler en mode hors-ligne ;
- donner aux développeurs une plus grande liberté, puisque leurs modifications ne peuvent affecter que leurs dépôts.

b) Fonctionnement

Pour bien saisir l'utilisation du *git*, définissons dans un premier temps quelque concepts importants.

Un dépôt est un espace où sont conservées toutes les modifications apportées au code source du logiciel.

Un commit est une structure de données permettant de sauvegarder les informations concernant les changements qui ont été apportés dans le code source. En outre, un commit contient d'autres informations importantes comme le nom du développeur, la date, l'état du projet ainsi qu'un message permettant d'éclaircir le but du commit.

Une branche est une fonctionnalité permettant d'avoir plusieurs versions de travail dans un même dépôt, sans que ces modifications entrent en conflits. Les branches permettent de séparer la version principale du programme des versions de travail, elles permettent aussi de travailler sur plusieurs fonctionnalités de manière simultanée.

Git est un logiciel en lignes de commandes, le tableau ci-joint illustre les différents commandes couramment utilisées :

git clone « adresse »	Effectue la copie locale d'un dépôt existant.
git status	Affiche le statut du fichier
git commit	Enregistre dans le système de gestion de version toutes les modifications ajoutées au projet.
git push	Envoi les commit sur un dépôt distant.
git fetch	Met à jour le dépôt local avec une version d'un dépôt distant. Cette mise à jour ne modifie pas l'état actuel du dépôt.

Figure II.3 : tableau exemples de commandes du *GIT*

c) Utilisation du *GIT* au cours du stage

Durant le stage, ce système de gestion de version a été utilisé pour le couplage. Nous avons créé un dépôt pour le couplage et les différents développements qui le composent. Ainsi, toutes les personnes qui souhaitent suivre le déroulement du projet pouvaient le faire en récupérant le dépôt. J'étais quant à moi chargée de créer de nouvelles versions au fur et à mesure que le projet avançait. La découverte de ce système de gestion de version a été un plus car je me suis vite aperçue que pour gérer un projet où plusieurs développeurs interviennent cela est une solution idéale.

III– Relation entre le stagiaire et les encadrants

Cette deuxième partie a pour objectif d'expliquer l'organisation pour communiquer entre tuteurs et stagiaires mais aussi entre les différents sites de l'INRA, chacun à l'origine d'un des deux environnements de travail.

1. Réunions

Afin de s'assurer du bon fonctionnement du stage, et pour permettre de réaliser le stage dans de bonne condition, assez régulièrement avaient lieu des réunions en présence de Ronan Trépos, et moi-même et quelques fois Robert Faivre. Ces réunions avaient le plus souvent lieu en salle de réunion de l'unité ou bien dans le bureau de monsieur Trépos.

Nous avons réalisé plusieurs réunions, la première a eu lieu le 8 avril 2013 afin de fixé le contexte du stage, d'exprimer les besoins et de donner les outils de travail.

La seconde réunion a eu lieu le 19 avril 2013, et avait pour objectif d'établir le planning prévisionnel.

La troisième réunion a eu lieu le 26 avril 2013 et consistait à présenter la fusion sur le modèle exemple. De cette réunion, nous avons mis au point la démarche de développement pour la généralisation du projet.

La quatrième réunion s'est déroulée le 17 mai pour discuter du code du couplage généralisé, c'est lors de cette réunion que nous avons évoqué la mise en place de l'interface graphique.

La cinquième réunion a eu lieu le 3 juin 2013, afin de faire le point sur l'avancée du projet. Et pour également parler de la journée des stagiaires et du résumé qui était à écrire.

La sixième réunion a eu lieu le 11 juin 2013, il s'agissait d'une répétition pour l'oral du jeudi 13 juin 2013, de présentation du stage à l'unité.

2. La visioconférence

Ce projet était en partenariat entre le réseau MEXICO et la plate-forme RECORD. Les développeurs des outils du réseau ne travaillant pas sur le centre de Midi-Pyrénées mais sur le site situé à Jouy-en-Josas, il a fallu organiser une visioconférence afin d'une part de se rencontrer et de fixer les modalités de travail. Pour réaliser ce rendez-vous, nous avons préparé les différentes questions pour résoudre les problèmes que nous rencontrions et nous avions au préalable envoyé les premiers développements. Cette visioconférence a eu lieu dans le bureau de Ronan Trépos, le 14 mai 2013. Nous avons abordé plusieurs aspects : dans un premier temps, le moyen de communication entre les deux équipes ; ainsi il a été choisi de convenir ensemble d'un jour de travail dans la semaine consacré au couplage. Le jour fixé est le vendredi. Dans un deuxième temps, nous avons abordé le mode de distribution du couplage : dans quel paquet sera intégré le couplage. Au sein du paquet d'exploration ou bien au sein du paquet de simulation de modèle ? La question méritait encore de la réflexion, mais la piste de créer un nouveaux paquet contenant le couplage, ne semblait pas convenir. Au cours de cette visioconférence, nous avons pu poser les questions que nous souhaitions afin de poursuivre le développement du projet.

3. Jour de travail

Comme convenue lors de la visioconférence tous les vendredis matins, nous communiquions avec monsieur Juhui Wang, par skype[7] afin de travailler ensemble sur le couplage. Ces rendez-vous, nous ont permis de résoudre différents soucis et ont permis à l'équipe de Jouy-en Josas de suivre le projet. Ce rendez-vous hebdomadaire nous a permis de garder le contact avec les développeurs du paquet d'exploration, et

[7] **Skype** est un logiciel gratuit permettant aux utilisateurs de communiquer via internet par messageries instantanée, appels téléphoniques ou encore visioconférence.

de fournir un travail d'équipe. Durant ces réunions, j'étais quelque fois la seule interlocutrice de MIAT, cela m'a permis de bien m'intégrer au projet et d'avoir une place au sein de l'équipe.

Bilan

Les travaux réalisés au cours de stage permettront aux utilisateurs de *rvle* de pouvoir effectuer des explorations de modèles. C'est un aspect important pour mieux comprendre les systèmes. Le coupage est réalisé et opérationnel, c'était l'objectif principal du stage. L'interface graphique est en cours de développement et mérite encore du temps de codage pour la rendre fonctionnelle. Ce stage est contributeur dans le réseau MEXICO car il permet d'étendre les fonctionnalités du paquet d'exploration *mtk*.

D'un point de vue personnel, ce stage m'a permis de connaître de nouvelles méthodes statistiques, notamment toutes les méthodes liées à l'exploration des modèles. Il m'a également permis de mettre en application mes compétences acquises durant mes trois années d'études post-bac. Il m'a apporté aussi l'opportunité de découvrir l'environnement dans le domaine de la recherche. J'ai apprécié cette expérience car je l'ai trouvée enrichissante pour mes connaissances personnelles.

Le travail que j'ai réalisé m'a permis de me perfectionner dans la programmation sous **R**. J'ai appris l'utilisation des paquets sous R, l'utilisation de différents méthodes. J'ai également acquis de nouvelles connaissances en informatique et en gestion de projet comme l'utilisation du système de gestion de versions. Ce stage est complémentaire à mon année d'étude au sein de la formation statistique et informatique décisionnelle. Il m'a apporté une nouvelle expérience, des nouvelles connaissances scientifiques et m'a permis de comprendre les enjeux d'un projet informatique.

Bibliographie - Sitographie

Livre

Analyse de sensibilité et exploration de modèles – applications aux sciences de la nature et de l'environnement.
R.Faivre, B.Ioos, S.Mahévas, D.Makowski, H.Monod, éd.
Edition Quae – 2013. 324 pages

Sites

Plate-forme RECORD : https://www6.inra.fr/record

Réseau MEXICO : http://reseau-mexico.fr/

Tutoriel paquet R shiny : http://rstudio.github.io/shiny/tutorial/

Site du cran : http://cran.r-project.org/

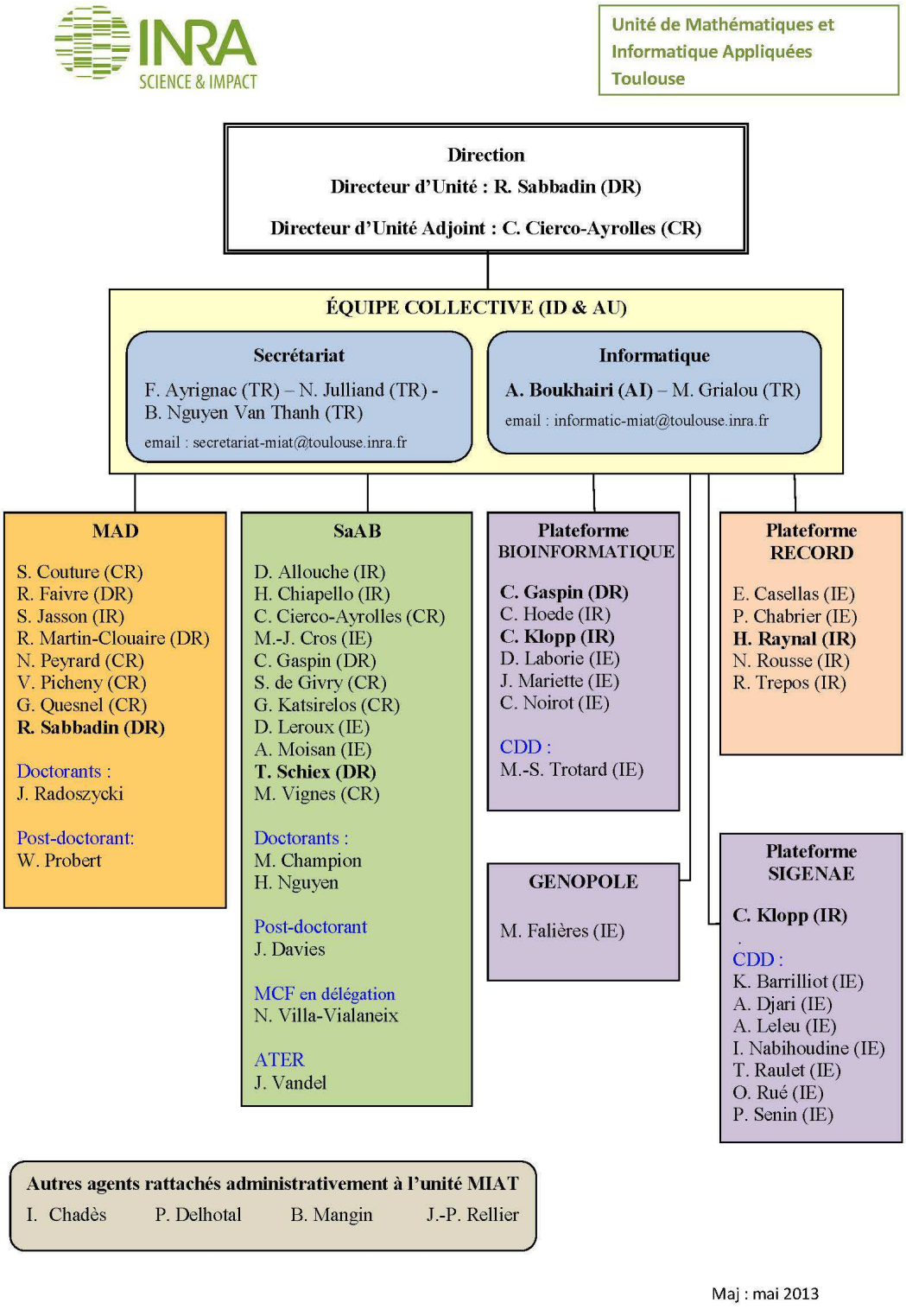
Annexes

Tables des annexes

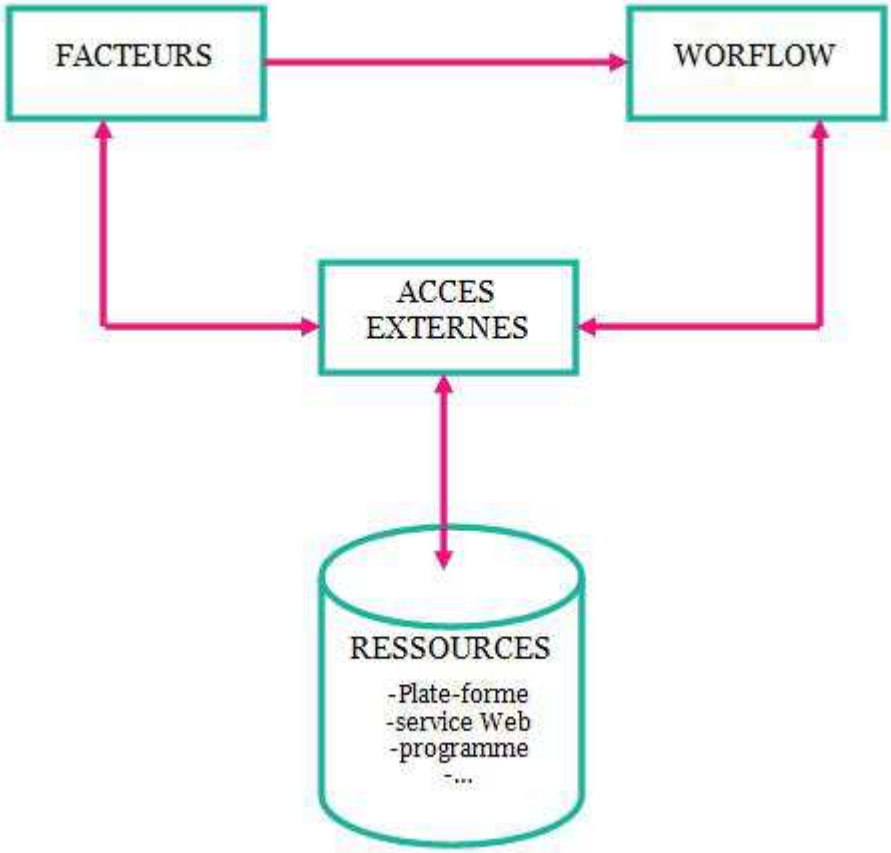
1 Exemple contenu de la variable « cv ».

```
VLE Model informations :
========================

Vpz location : file = 2CV-exploration.vpz , pkg =
tp5_4_correction

Experiemental condition settings and default value :
* condAGB.hiMax = 0,506732 <numeric>
* condAGB.r1Rue = 1 <numeric>
* condAGB.r1hi = 19,42438 <numeric>
* condAGB.r2Rue = 1 <numeric>
* condAGB.r2hi = 0,8 <numeric>
* condAGB.rateHI = 0,015 <numeric>
* condAGBVar.mode = port <character>
* condAGBVar.variables = list("AGBiomass"), list("Yield", 0),
list("HIpot"), list("HI"), list("dayCount") <VleSET>
* condAvailableWater.mode = port <character>
* condAvailableWater.name = sum <character>
* condAvailableWater.unit = day <character>
* condAvailableWater.variables = list("Infiltration", 0),
list("Irrigation", 0), list("IncomingRunOff", 0), list("RunOff",
0) <VleSET>
* condBilan.coutEau = 0,76 <numeric>
* condBilan.prixVente = 106,71 <numeric>
* condDecAgent.apportEau = 30 <numeric>
* condDecAgent.nbJoursAvantRetour = 9 <integer>
* condDecAgent.quantiteEauPeriodeSeche = 10 <numeric>
* condDecAgent.quantiteEauSolPortant = 10 <numeric>
* condDecAgent.stockEau = 300 <numeric>
```

2 Organigramme de l'unité MIAT en mai 2013.



3 Architecture général du paquet d'exploration *mtk*.



4 Code de la fonction « .simule » du couplage générique.

```
RvleMtk.simule<-function(X ,pkgVle , modelVle , view ,
                         indiceLigneVue , nomVarSortie ,fact)
{
  library(rvle)
  # Appel de RVLE pour générer le modèle
  m<-new("Rvle",file=modelVle,pkg=pkgVle)
  inValues=data.frame(X)

  # Modification type de variables
  nameList<-c()
  nbFact <- length(fact)
  for (i in 1:nbFact){
      nameList<- c(nameList,getName(fact[[i]]))
      type<-getType(fact[[i]])
      if (type == "numeric"){
            inValues[[i]]=as.double(inValues[[i]])
            }
      if (type == "integer"){
            inValues[[i]]=as.integer(inValues[[i]])
            }
      if (type == "character"){
            inValues[[i]]=as.character(inValues[[i]])
            }
      }
  names(inValues)=nameList

  # execution et recuperation des informations
  oldPluginView = getDefault(model,"outputplugin")[[view]]
  rvle.setOutputPlugin(model@sim, view,"vle.output/storage")
      s = rvle::run(model,outputplugin= c(vueFinalBilan =
"vle.output/storage"), plan = "linear", inputs=inValues)

  r = results(s)
  rvle.setOutputPlugin(model@sim, view, oldPluginView)

  varSortie=c()
  for (i in 1:nrow(inValues)){
            y=r[[i]][[view]][indiceLigneVue,nomVarSortie]
            varSortie=c(varSortie,y)
            }

  Y = as.data.frame(varSortie)
  output=list(main=Y, information=list(package=pkgVle,
       model=modelVle, vueSortie=view, indiceLigneVue=
       indiceLigneVue, nomVarSortie=nomVarSortie))

return (output)}
```

5 Couplage générique : déclaration d'un facteur.

```
fact_apportEau <- make.mtkFactor (
                  name="condDecAgent.apportEau",
                  type="numeric",
                  distribName="unif", nominal=30,
                  distribPara=list(min = 5, max = 50))
```

6 Couplage générique : création du modèle dans *mtk*.

```
#création du modèle
mtk.evaluatorAddons(where="~/rvleAddons.R", name="rvle", main
= "RvleMtk.simule", authors="Maeva", summary=NULL, plot=NULL,
print=NULL)

#chargement du modèle
source("~/mtkrvleEvaluator.R")
```

7 Couplage générique : création du processus de simulation.

```
cv.evaluate <- mtkNativeEvaluator(model="rvle",
                information=list(pkgVle="tp5_4_correction",
                modelVle="2CV-exploration.vpz",
                view="vueFinalBilan",indiceLigneVue=1,
                nomVarSortie="model:Bilan.Profit",Facteur)
)
```

8 Maquette de l'interface.