Il est à noter que dans le cadre du développement d'un modèle de simulation, l'implémentation du modèle et de son interface graphique sont étroitement liées. D'une part, la plateforme de modélisation choisie contraint fortement le type et la diversité des représentations. Gama et GeoMASON, par exemple, proposent des modes de visualisation de données géographiques bien plus avancés que NetLogo ou Repast. L'interface graphique développée pour chaque modèle est donc largement influencée par la plateforme de simulation dans laquelle il est exprimé. D'autre part, dans la plupart de ces plateformes de simulation, l'interface graphique est implémentée au même niveau que le code-source du modèle en lui-même. Dans Netlogo et Gama par exemple, l'interface graphique est programmée directement dans le modèle en lui-même, en se basant sur les variables qui y sont déclarées. Il n'est donc pas possible de créer une interface graphique générique au sein des plateformes de simulation, laquelle pourrait s'appliquer à plusieurs modèles différents. Il est nécessaire, pour chaque modèle, de reconstruire l'interface depuis les briques de bases proposées par les plateformes, c'est-à-dire un ensemble de primitives graphiques permettant de composer une interface intégrée au modèle.

Dans l'exploration de SimFeodal, la création en direct de quelques graphiques correspondant à des indicateurs étudiés permet d'assurer un rôle de filtrage, grossier, à l'exécution d'une expérience, c'est-à-dire de l'ensemble des réplications nécessaires à son évaluation. Après une modification du code-source du modèle, et avant de lancer de nombreuses simulations, on exécute quelques simulations « manuellement ». On s'assure alors, en direct, que les indicateurs affichés ne présentent pas de caractères aberrants. Cela permet de vérifier, avant de lancer des calculs plus conséquents, que le déroulement de la simulation semble cohérent, c'est-à-dire, le plus souvent pour des modifications mineurs du modèle, qu'un *bug* n'a pas été introduit involontairement.

Le recours à ce type de visualisation en direct des simulations ne peut toutefois être généralisé, c'est-à-dire sorti de son rôle de pré-filtre, en raison de deux principales contraintes.

La première contrainte, déjà évoquée plus haut, est que le modèle SimFeodal est fortement stochastique. Dès lors, la visualisation des indicateurs d'une simulation particulière n'est pas suffisante pour estimer le comportement du modèle. En conséquence, les indicateurs choisis pour l'évaluation de SimFeodal prennent presque tous en compte la variabilité des résultats induite par l'exécution de réplications. Certains environnements techniques[14] permettent toutefois de mener concomitamment plusieurs réplications d'un même modèle et de visualiser directement pendant l'exécution les résultats agrégés des réplications. Cette première contrainte, liée au besoin d'analyser des réplications plutôt que des exécutions isolées, peut donc être dépassée en adaptant l'implémentation du modèle pour faire usage de ces capacités de multi-simulation.

La seconde contrainte est cruciale dans le cas de SimFeodal et invalide l'usage des méthodes de visualisation en direct. On l'a vu, l'exploration des

14. Gama dans sa dernière version (1.8) par exemple, voir https ://gama-platform.github.io/wiki/RunSeveralSimulations.html.

sorties de simulation du modèle repose sur la consultation systématique de plusieurs indicateurs, dont le nombre peut se révéler important pour une analyse approfondie.

Tout d'abord, il est concrètement difficile de représenter tous ces indicateurs au sein de l'interface graphique d'une plate-forme de simulation agent, comme on peut le remarquer, par exemple en figure 5.1 : l'espace occupé par les quelques indicateurs temporels et numériques affichés est déjà important et rend l'interface d'ensemble complexe. De plus, et c'est sans doute le verrou majeur, la temporalité de l'exécution d'une simulation – ou même des réplications nécessaires – est bien plus courte que celle requise pour la compréhension des résultats produits. Une simulation requiert ainsi au maximum quelques minutes. Pour pouvoir examiner tous les indicateurs pendant cette durée, il serait nécessaire de mettre la simulation en pause régulièrement, presque à chaque pas de temps. On disposerait alors d'un temps suffisant pour observer les indicateurs (organisés en onglets dans l'interface visible dans la figure 5.1). L'analyse des indicateurs de sortie de simulation demandent en effet un examen approfondi, et non simplement superficiel, pour pouvoir juger de l'adéquation de ce que ces indicateurs représentent vis-à-vis des attentes thématiques.

Cette contrainte est renforcée par la pratique d'évaluation que la plupart des chercheurs mobilise. L'évaluation n'est pas une étape unique et finie, il est utile de pouvoir revenir sur les résultats à différents moments. Cela est par exemple nécessaire quand il s'agit de comparer de nouveaux résultats produits à ceux générés par des expérimentations antérieures. On ne peut alors se contenter d'évaluations en direct, même en y consacrant un temps important, simplement parce que par nature, ces évaluations seront à reproduire en plusieurs occasions, et qu'il ne serait alors pas rationnel de relancer, à chaque fois, de nouvelles simulations correspondant à des configurations de paramètres et de mécanismes déjà éprouvées.

Un dernier élément contribue à la difficulté de se baser sur une évaluation en direct : en plus du chercheur, amené à revenir de multiples fois sur les résultats d'une expérience, un modèle co-construit peut être évalué par plusieurs chercheurs différents. C'est d'autant plus fréquent en situation d'interdisciplinarité, où les points de vue de chacun des membres sont complémentaires et nécessaires. Sauf à faire preuve d'une discipline exacerbée, par exemple en réalisant l'ensemble du travail d'évaluation uniquement en séances de travail simultanées et collectives, l'évaluation par plusieurs personne demande que chacun puisse mener ces analyses selon ses propres temporalités. En choisissant de baser l'évaluation d'un modèle uniquement sur une analyse en direct, il faudrait alors que chaque chercheur, à chaque fois qu'il souhaite évaluer une même expérience, ré-exécute le modèle de nombreuses fois. Cela serait naturellement possible, mais constituerait clairement un gâchis certain de ressources.

L'évaluation d'un modèle interdisciplinaire et exploratoire ne peut donc que difficilement être réalisé en direct, qui plus est quand elle demande de faire appel, dans un cadre de co-construction, à plusieurs points de vue hétérogènes.

Les modalités mêmes de l'exploration des sorties d'un modèle à évaluation visuelle requièrent donc que les indicateurs soient visibles et explorables à des temporalités différentes, par des chercheurs différents, depuis des lieux divers.

Il est donc indispensable que les indicateurs soient enregistrés et consultables simplement à tout moment, ce qui élimine de fait la visualisation des indicateurs en direct pendant l'exécution des simulations comme unique méthode d'exploration du comportement des modèles exploratoires et descriptifs.

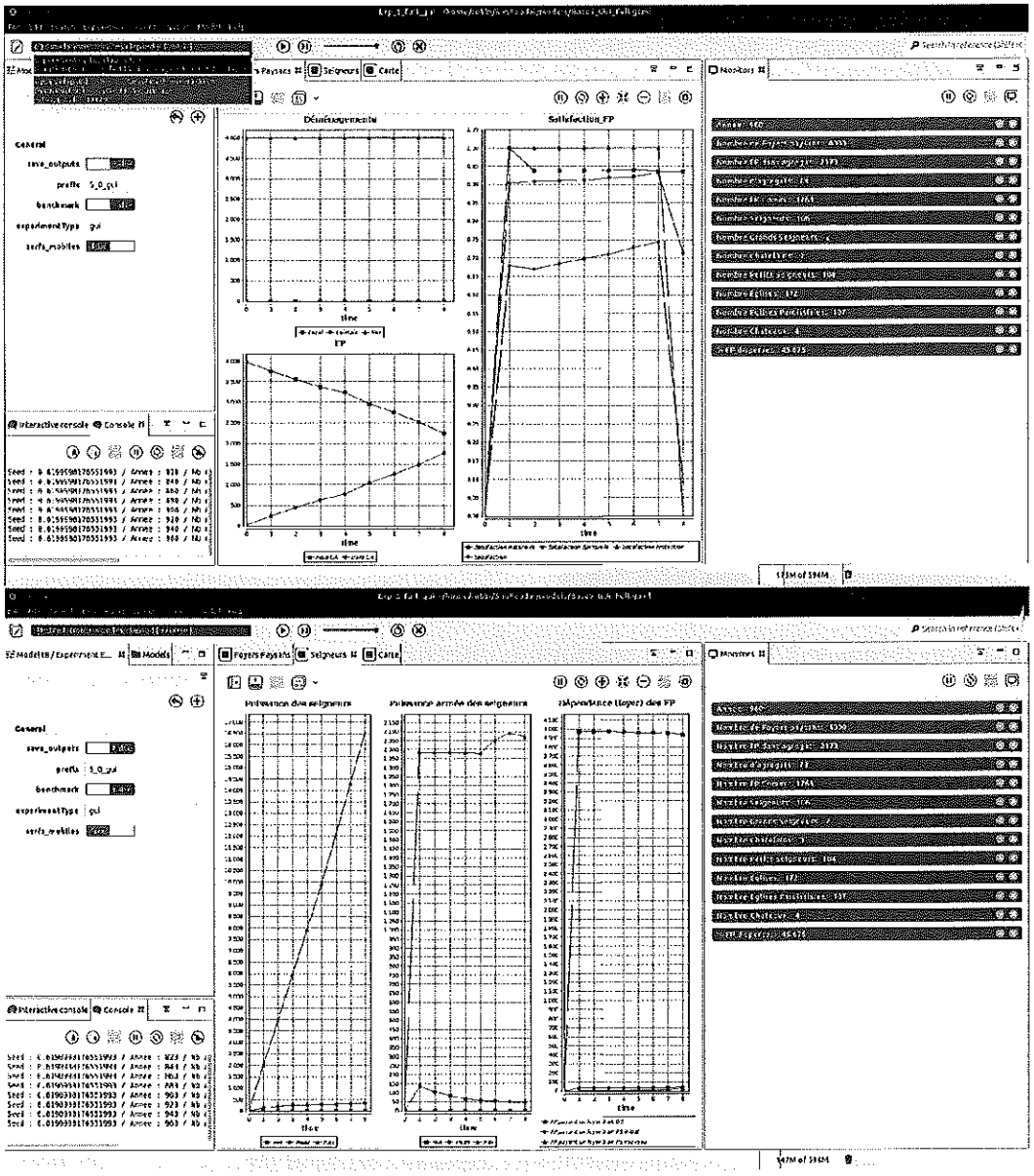


FIGURE 5.1 – Indicateurs intégrés à l'interface graphique interne de SimFeodal. Dans ces captures d'écrans, il s'agit d'indicateurs liés aux foyers paysans et aux seigneurs.

La visualisation en direct n'est donc pas mobilisable en tant que méthode d'évaluation principale, mais elle peut tout de même, comme dans un usage très classique, être utilisée comme un outil de validation interne pour tester chaque modification dans les valeurs de paramètres, remplissant alors le rôle de « préfiltre » décrit auparavant. Visualiser une seule simulation, avant d'en exécuter les réplications nécessaires, permet ainsi déjà de vérifier que les modi-

fications apportées dans les valeurs de paramètre ou dans les mécanismes n'ont pas entraîné l'apparition de *bugs* ou d'incohérences immédiatement visibles.

Nous avons donc choisi de développer une interface graphique, très sommaire mais permettant des allers-retours rapides entre l'implémentation et l'exécution, au sein de l'implémentation de SimFeodal. Cette interface n'affiche qu'un nombre réduit d'indicateurs (Figure 5.1), ainsi qu'une représentation cartographique (Figure 5.2) utile à une analyse rapide du comportement d'ensemble du modèle.

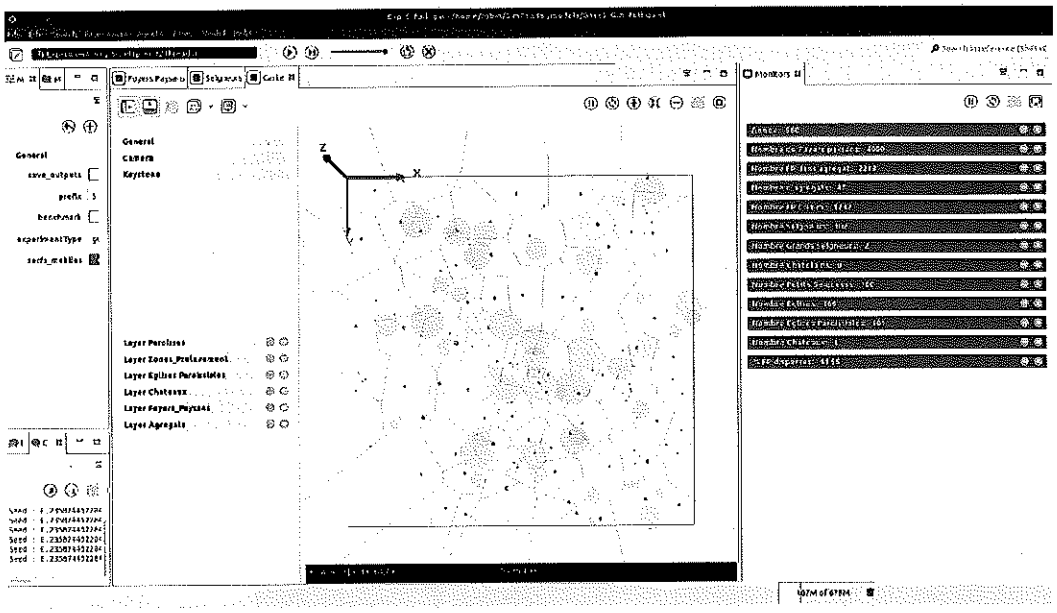


FIGURE 5.2 – Visualisation intégrée à l'interface graphique interne de SimFeodal : cartographie synthétique de l'espace modélisé.

Ces indicateurs intégrés à l'interface graphique interne de SimFeodal permettent de mener une étape préalable à l'évaluation du modèle. Cette étape vise à effectuer un premier filtrage des simulations avant d'exécuter les expériences en elles-mêmes. L'ajout de cette interface graphique vient donc renforcer l'outillage d'évaluation de SimFeodal. A cette étape, il manque encore un outil véritablement adapté à l'analyse conjointe des réplications du modèle, a posteriori de l'exécution des nombreuses réplications.

5.2.2 Générer les indicateurs

La production des indicateurs doit nécessairement être réalisée en aval de l'exécution des simulations (« *offline* » dans GRIGNARD et DROGOUL 2017). Il faut pour cela disposer d'outils adaptés au traitement des données produites, c'est-à-dire répondant aux contraintes identifiées auparavant (sous-section 5.1.4). La contrainte principale est d'être en mesure de gérer la masse de données produites. On l'a vu, cela élimine d'office les outils de type tableurs, ou encore les outils de manipulation graphique de données les plus courants. Pour les raisons évoquées dans le chapitre 1 (Positionnement : pourquoi utiliser des outils libres ?), seules les solutions techniques libres étaient envisageables.

Certains outils graphiques, basés sur des logiciels libres en arrière-plan

(PSPP, R Commander, Orange), sont extrêmement aisés à prendre en main et auraient pu constituer un bon choix. Pourtant, avec une trentaine d'indicateurs à produire pour chaque expérience, donc de manière régulière, nous avons préféré nous tourner vers des outils plus orientés vers une interface en ligne de commande (*Command Line Interface*, abrégés *CLI*).

L'utilisation de CLI a plusieurs intérêts gravitant autour de la reproductibilité des traitements. En premier lieu, ils permettent une adaptation aisée et rapide aux différents jeux de données. Ainsi, partant du principe que les données générées par les réplications et expérimentations sont de même structures, il suffit généralement de modifier le chemin d'entrée des fichiers résultants pour reproduire à l'identique une analyse sur un nouveau jeu de données.

De manière plus technique, on peut remarquer que les différents indicateurs de sortie de simulation choisis présentent souvent des caractéristiques communes, aussi bien dans le traitement nécessaire que dans les formats (graphiques) produits.

Par exemple, la grande majorité des indicateurs reposent sur une première agrégation des données par réplication et pas de temps simulé, puis par une seconde agrégation montrant la variabilité des situations générées, au niveau de l'expérimentation[15]. En terme de manipulation de données, seuls le calcul de la variable à mobiliser, et éventuellement l'agent caractérisé – la variabilité du nombre de foyers paysans et la variabilité du nombre d'agrégats ne diffèrent que par le type d'agent sur lequel le calcul est effectué –, sont ainsi à adapter dans ces nombreux indicateurs de sortie. Les variations, en terme de code-sources, sont donc le plus souvent des adaptations minimes (nom de l'agent, type d'agrégation...). Le recours à des traitements en *CLI* permet ainsi un simple copier/coller, voir la création de fonctions dédiées, pour effectuer ces traitements très récurrents.

Au niveau des sorties graphiques, on peut aussi remarquer que la structure des graphiques, en elle-même, est assez largement identique : on représente les pas des temps (les années simulées) en abscisse, un indicateur statistique en ordonnée, et la variabilité sous la forme de *box-plot* minimalistes (« *minimal boxplot* », promus par Edward Tufte pour minimiser le ratio données-encre (TUFTE 2001, p. 123-125)). En disposant d'un environnement de type *CLI*, et qui plus est en faisant usage de solutions graphiques construites sur une syntaxe régulière et générique (voir , section 5.4.2.1), il devient très confortable de générer les différents indicateurs de sortie souhaités, puisqu'il suffit d'adapter les graphiques déjà conçus.

Avec ces solutions logicielles d'analyse de données et de visualisation, il est facile de concevoir et d'implémenter les codes informatiques nécessaires

15. On peut considérer ces agrégations comme une succession d'opération imbriquées : pour montrer l'évolution d'un indicateur tel que le taux de foyers paysans dispersés au cours du temps simulé, il faut (1) calculer le ratio entre nombre de foyers paysans dispersés et nombre total de foyers paysans, (2) pour chacun des pas de temps simulé, (3) pour chaque simulation, (4) pour l'ensemble des réplications d'une expérience, (5) éventuellement pour chacune des expériences d'une phase plus large d'expérimentation qui ferait varier des valeurs de paramètres.

à la génération des indicateurs de sortie de simulation. De plus, l'exécution de ces programmes est extrêmement rapide : les différents fichiers de sortie de simulation sont lus et parcourus une unique fois pour en tirer toutes les variables nécessaires à l'établissement des indicateurs.

En ayant choisi de mener une évaluation *a posteriori* – plutôt qu'en direct – basée sur l'observation d'indicateurs générés – par des outils adaptés au traitement de données massives – de manière automatisée, on dispose donc, pour chaque expérience, d'un ensemble de fichiers numériques : chacun des indicateurs de sortie est contenu dans un fichier unique, dans un format facilement exploitable et ré-utilisable.

5.2.3 Organiser les indicateurs en rapports paramétrables

Du point de vue de la manipulation, la création de fichiers informatiques indépendants correspondant aux différents indicateurs de sortie de simulation est extrêmement pratique : on peut facilement les identifier, les transférer et les adapter, par exemple pour en rendre le contenu plus compréhensible par un public différent.

En revanche, du point de vue de la comparaison des résultats, cette forme n'est pas la plus adaptée. Si l'on peut facilement comparer un même indicateur portant sur deux expériences différentes, la tâche se complique quand il s'agit d'avoir une vision globale des différences dans les indicateurs entre deux expériences. Pour cela, la démultiplication des fichiers correspondant aux indicateurs se révèle rapidement être un obstacle : l'utilisateur est en effet amené à jongler entre de très nombreux fichiers.

Les rapports comme instruments de comparaison Pour faciliter la comparaison d'indicateurs multiples – et d'une forte diversité –, il est nécessaire de les organiser au sein d'une structure englobante. Nous entendons ici par organisation, une présentation structurée, suivant un certain ordre, identique selon les expériences, adaptée à une évaluation des résultats. Pour cela, nous avons choisi d'organiser les indicateurs de sortie de simulation au sein de « rapports ». Cela permet, même en présence de nombreuses expériences, de rassembler l'ensemble des indicateurs de sortie propres à chacune dans un unique fichier, à la structure toujours similaire. Un premier apport, majeur, concerne l'archivage des sorties de simulation. Avec des rapports comprenant l'ensemble des indicateurs de sortie de chaque expérience d'un modèle, il est ainsi simple de conserver des traces de l'ensemble des versions et sous-version d'un modèle. Cela permet de garantir une certaine pérennité à ce modèle, à sa documentation, et ainsi simplifie le travail rétrospectif de caractérisation de l'évolution d'un modèle.

L'intérêt majeur de la structuration en rapports est surtout de faciliter la comparaison des expériences, c'est-à-dire des indicateurs de sortie des différentes expériences. On peut ainsi, par exemple, placer côte à côte, visuellement, deux rapports rendant compte de deux expériences différentes, et, en les faisant défiler simultanément, comparer point par point, c'est-à-dire indicateur

par indicateur, leurs résultats respectifs, de manière visuelle et intuitive.

Les formes que peuvent prendre des rapports, tout autant que les modalités de leur production, sont multiples et extrêmement diverses. La forme la plus simple et courante consiste à produire manuellement le rapport en insérant les indicateurs adaptés au fur et à mesure, par exemple dans un traitement de texte. A l'opposé, on peut noter les possibilités de créations entièrement automatisées de rapports complets, comprenant par exemple des descriptions et commentaires textuels générés à la volée, en fonction d'expressions conditionnelles [16].

Pour SimFeodal, nous avons choisi de restreindre au maximum la manipulation manuelle. On souhaitait générer un rapport entièrement automatique, ne requérant pas d'action spécifique en dehors du choix des données depuis lesquelles créer les indicateurs. A l'inverse, on a choisi de ne pas ajouter de fonctionnalité de commentaire automatique des indicateurs de sortie : la richesse – et la difficulté– d'une approche interdisciplinaire telle que la notre est constituée par la multiplication des analyses et points de vue. Il n'y avait donc aucun besoin de générer des commentaires standardisés et automatiques, forcément moins aboutis que les commentaires de chacun des co-concepteurs du modèle. Le rapport produit (figure 5.3) n'intègre donc que les indicateurs, sous forme de tableaux et de graphiques. Ces indicateurs sont organisés par partie, en l'occurrence en fonction du type d'entités et de comportement qu'ils décrivent.

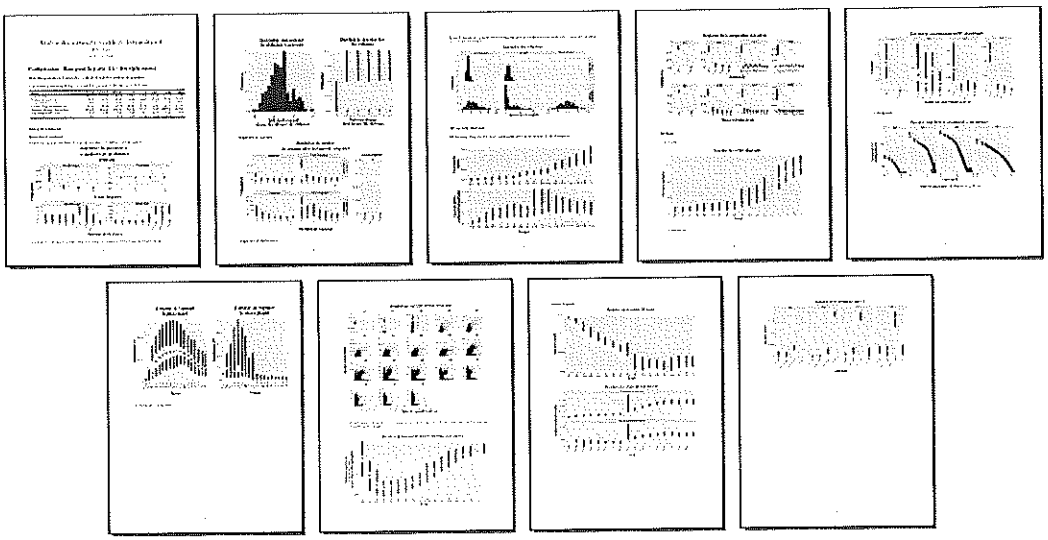


FIGURE 5.3 – Un exemple de rapport automatique généré pour une expérimentation (étape 0) de SimFeodal. La version en taille réelle est reproduite en annexe X.

16. Voir par exemple l'application « SOFIE » de l'Observatoire des Territoires du Commissariat général à l'égalité des territoires (CGET), qui génère automatiquement des commentaires relatifs aux inégalités femmes/hommes dans l'accès à l'emploi. Les commentaires propres aux types d'inégalités majeures de chaque EPCI sont produits de manière automatique depuis les données. http ://outils.observatoire-des-territoires.gouv.fr/sofie/

Structurer des rapports pour aller vers la reproductibilité des analyses On a donc fait le choix de se baser sur des rapports automatisés et minimalistes, ne contenant que les indicateurs dans une forme structurée. Ce choix s'appuie sur des raisons multiples, qui ont toutes en commun une recherche de reproductibilité des résultats et des analyses menées. Une reproductibilité théorique (encore une ref au positionnement), puisque les résultats de simulation doivent pouvoir être analysés et reproduits par des chercheurs potentiellement intéressés. Mais cette recherche de reproductibilité est aussi pratique, rendue aussi nécessaire par les méthodes de modélisation suivies. Tel qu'explicité auparavant (sections 5.1.2 et 5.1.3), celles-ci s'appuient sur de nombreux allers-retours, ce qui requiert une capacité constante à reproduire et à affiner des résultats déjà observés.

La quantité d'expériences requises pour arriver à un état satisfaisant du modèle est tributaire de ces allers-retours, et le nombre de rapports qu'il faut pouvoir produire est important. La fréquence de production de ces rapports est forte, et le modélisateur a alors tout intérêt à en fluidifier et accélérer le processus de création.

Dans une telle situation, la création d'un rapport automatisé garantit une calcul et une production simplifié et rapide des indicateurs sur les nouvelles données. Cela permet un examen des sorties de simulation presque immédiatement après leur exécution. Cette automatisation permet aussi de mener une seconde évaluation – après le pré-filtrage constitué par l'observation d'indicateurs en direct – du bon déroulement « interne »[17]. Le caractère fixe d'un rapport automatisé se base ainsi sur une structure de données contraignante, par exemple constitués en n fichiers dotés de plusieurs colonnes spécifiquement attendues. Les caractéristiques de ces données sont elles-mêmes contraignantes. Un rapport automatisé ne fonctionne, par exemple, qu'en présence d'un nombre pré-défini de réplications complètes. En l'absence d'un de ces critères dans des données en sortie de simulation, le rapport ne peut être généré et émet une erreur. Par exemple, si le nombre de réplications est plus faible qu'attendu, ou encore si tel attribut d'un agent a changé de type informatique, la création du rapport échoue. La présence ou non de cette erreur constitue donc un nouveau filtre de vérification de la validité du modèle. Cela permet, là encore, de détecter des simulations qui présenteraient des comportements incomplets ou aberrants en terme de production de données.

Un autre intérêt majeur des rapports, déjà pointé en avantage des outils de type *CLI* est leur adaptabilité. On a vu (chap. 3) que les indicateurs à examiner sont nombreux et surtout, évolutifs, dans le sens où ces indicateurs ont fortement été modifiés, remplacés, affinés, au cours des étapes de paramétrage de SimFeodal. L'utilisation de rapports automatiques permet de minimiser le nombre de modifications à effectuer en cas de changements d'indicateurs. Le programme informatique qui génère les rapports s'appuie ainsi sur un code-source unique, générique aux simulations de chaque modèle. Lors d'un chan-

17. Au sens de l'évaluation interne, c'est-à-dire du bon fonctionnement, exempt de *bugs*, du modèle de simulation implémenté.

gement d'indicateurs, il suffit alors de modifier ce code-source en une seule place, et tous les appels à ce programme seront alors modifiés en conséquence. À partir de là, pour mettre à jour l'ensemble des rapports déjà produits, c'est-à-dire regroupant les indicateurs de chacune des expériences passées, il suffit de ré-exécuter la routine de production des rapports.

Dans le cas de SimFeodal, caractérisé par de fréquents changements dans la forme et le calcul des indicateurs, cela a représenté un gain de temps et d'efficacité très conséquent. Par exemple, lors de certaines phases de paramétrage, on pouvait être amenés à faire évoluer le modèle quotidiennement et à tester, à cette même fréquence, plusieurs jeux de paramètres. Il fallait donc analyser les résultats de plusieurs expériences chaque jour, et régulièrement ajouter des indicateurs graphiques afin d'affiner l'évaluation. Ces indicateurs devaient aussi être ajoutés aux expériences des jours précédents, et au final, on re-générait parfois jusqu'à une plus d'une dizaine de rapports sur des cycles temporels courts de quelques jours.

Par extension, cette même démarche d'automatisation, basée sur l'utilisation d'outils de type *CLI*, devrait pouvoir s'appliquer à l'identique, avec les mêmes avantages, dans le cadre plus large de l'évaluation visuelle de modèles (ref chap 3, section 3.1.4).

Dépasser les limites de la compatibilité d'ensemble En réalité, la démarche de reproductibilité des rapports constitue plus un objectif théorique qu'un stade, « reproductible » qu'il serait possible d'atteindre. Il s'agit ainsi plus de s'inscrire dans la voie de la reproductibilité que de chercher à parvenir à cet objectif largement inaccessible dans les faits.

Comme explicité dans l'encadré 3.2 sur l'incrémentalité des indicateurs, une limite forte empêche d'atteindre une reproductibilité absolue des analyses du comportement des différentes versions de SimFeodal. Les données générées par les différentes versions du modèle ne sont en effet pas systématiquement « compatibles ». On entend par là qu'elles ne présentent pas toute exactement la même structure, à commencer par les variables enregistrées. Quand bien même il aurait été choisi dès le départ d'enregistrer le plus de sorties possibles, la reproductibilité de l'analyse échoue sur les données produites par le modèle et ses nombreuses versions : le modèle évolue, et avec lui, certaines variables apparaissent et d'autres deviennent caduques. La structure contraignante et précise des données nécessaires à la génération des rapports ne peut être entièrement satisfaite. La prise en compte de l'évolution du modèle demande une adaptation régulière – mais aussi rare que possible – des programmes qui génèrent ces rapports.

On ne peut donc satisfaire globalement à un objectif de reproductibilité, mais il est toutefois possible de limiter la déviance à cette ambition. Pour cela, on peut agencer les différentes versions du modèles au sein de « générations » de modèle, c'est-à-dire d'ensembles de versions présentant des attributs comparables et générant des données de même structure. Plutôt que d'adapter le code-source des rapports à chaque nouvelle version du modèle, ou encore de

ne jamais l'adapter et donc d'être tributaire de la structure des toutes premières versions du modèle, cela constitue un choix intermédiaire qui permet de limiter le nombre de variantes de rapports. Cette approche suit les grandes lignes du développement logiciel général. Les itérations successives d'un logiciel sont constituées de versions « majeures » – les générations de modèles dans notre cas –, qui n'assurent pas nécessairement de compatibilité avec les versions majeures précédentes, et de versions « mineures », dans lesquelles la compatibilité est assurée [18].

Pour revenir aux rapports voués à l'évaluation d'un modèle, en inscrivant les différentes versions du modèle – et des programmes générant les rapport correspondant – dans des sous-ensemble de versions, les « générations », les différents rapports peuvent être considérés comme reproductibles et automatiques au sein de ces générations. Pour SimFeodal, cela implique d'organiser les différentes versions du modèle – résultant des étapes de paramétrage, cf. chap4 – au sein de grandes générations, à chaque changements structurels des mécanismes ou données produites par le modèle.

Ajouter un schéma avec les générations, versions et sous-versions

Les rapports, des instruments suffisants ? A l'issue de la conception et de l'implémentation de ces rapports automatiques, on dispose donc, pour chaque expérience, d'un document aisément partageable et lisible. Ces documents s'enrichissent, au fur et à mesure des générations de modèles, de nouveaux indicateurs, et sont comparables au sein de ces générations. Cela pourrait constituer la dernière étape de la création d'outils d'évaluation d'un modèle, dans la limite d'un nombre de versions ou de génération de modèles assez restreint.

SimFeodal, comme c'est souvent le cas dans les modèles à base d'agent, a toutefois été caractérisé par une forte quantité d'allers-retours entre le modèle et ses résultats, entraînant à chaque fois de nouvelles expérimentations (cf. section 5.1).

On a vu que la manipulation d'un grand nombre d'indicateurs, même pour une quantité restreinte d'expériences, disqualifiait l'usage de fichiers individuels et poussait à l'usage de rapports structurés. Avec un grand nombre d'expériences, les mêmes limites apparaissent pour les rapports : la masse d'expériences rend partiellement caduque l'utilisation unique des rapports automatiques. Il est en effet aisé de comparer, sur un même écran d'ordinateur, deux ou trois rapports, mais dès lors qu'il faut en comparer un plus grand nombre, la manipulation conjointe des rapports devient complexe, tout autant que d'avoir une vision globale des résultats principaux de chaque expérience.

18. Par exemple, un fichier de dessin vectoriel créé avec le logiciel Adobe Illustrator 15.0 ne sera pas lu correctement avec une version 14.0. Ce fichier présentera toutefois un compatibilité parfaite avec les versions 15.1 à 15.*n* du logiciel.

5.2.4 Organiser les rapports : Dashboards

Pour être en mesure de comparer de nombreux éléments, il est nécessaire de passer d'une exploration linéaire, fondée sur le visualisation successive de chacun des indicateurs, à une exploration globale et interactive. En pratique, plutôt que de faire défiler visuellement les nombreuses pages d'indicateurs, mieux vaut une interface présentant les points clefs de l'évaluation et qui permette d'entrer dans le détail de chacun des indicateurs dans un second temps, sur demande. Comme le résume le « mantra » de l'analyse visuelle (« *Visual information-seeking mantra* ») de Ben SHNEIDERMAN :

> « Overview first, zoom and filter, then details on demand »
>
> SHNEIDERMAN 1996, p. 2

5.2.4.1 Les *dashboards*

Cette logique, assez universelle désormais, est celle qui préside à la création des nombreux « tableaux de bord », ou « *dashboards* » que l'ont voit émerger depuis la fin des années 1990. Rob Kitchin et ses co-auteurs définissent ainsi les *dashboards*, en s'appuyant sur les travaux de Few notamment :

> « For Few [(FEW 2006a, p.34)] a 'dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance'. Just as a car dashboard provides critical information needed to operate the vehicle at a glance, indicator dashboards provide key information for running companies or cities (~~Dubriwny & Rivard, 2004~~)[(RIVARD et COGSWELL 2004)]. »
>
> KITCHIN, LAURIAULT et MCARDLE 2015, p. 11

Très répandus dans le monde de l'informatique décisionnelle (*Business Intelligence, BI*), ces outils permettent d'explorer des données d'entreprises, par exemple des résultats financiers. Pour ce faire, ils mettent en avant, dans une interface unique, des indicateurs clés (*Key Performance Indicators, KPI*), qu'il est ensuite possible de filtrer et d'affiner, par exemple par sélection de différents intervalles temporels.

Les *KPI* jouent le rôle d'indicateurs synthétiques, c'est-à-dire qu'ils s'adressent à des gestionnaires, par exemple des *managers*, qui ont une expertise importante sur les résultats produits. Les utilisateurs des *dashboards* ne sont donc pas des analystes, à même d'explorer eux-mêmes les données mobilisées, mais plutôt des thématiciens qui se fondent sur les indicateurs présentés pour prendre des décisions.

La dichotomie « analyste/décisionnaire » s'exprime aussi dans le domaine de la recherche, et notamment dans la recherche en géographie urbaine à visée applicative. Avec l'avènement des données massives et de leur prise en compte pour la gestion des villes (*smart cities*), les géographes se sont aussi penchés sur des outils de ce type. En résulte une utilisation de plus en plus fréquente de

dashboards en géographie urbaine (« *city-dashboards* », ROUMPANI, O'BRIEN et HUDSON-SMITH 2013 ; KITCHIN, LAURIAULT et MCARDLE 2015 ; BATTY 2015).

Le parallèle avec le monde de l'informatique décisionnelle est en effet présent dans les types d'utilisateurs et de producteurs de ces outils. Il s'agit de mettre à disposition d'experts thématiques (les décideurs publiques) des indicateurs clés, issus de calculs parfois complexes. Cela afin de leur permettre d'évaluer une situation donnée et de prendre les décisions politiques adéquates.

Le constat ayant mené à l'apparition des premiers *dashboards*, tant en informatique décisionnelle qu'en géographie urbaine, est identique. Les informations nécessaires à l'évaluation d'une situation (financière, relative aux politiques publiques...) sont de plus en plus nombreuses et hétérogènes. Les indicateurs permettant de mener ces évaluations, pensés pour les décideurs qui en feront usage (*managers*, acteurs publiques...), se démultiplient et se diversifient aussi par conséquence.

Inspiré autant par l'usage *BI* que par l'usage géographique, nous considérons que ces outils peuvent se révéler utiles dans l'évaluation de modèles de simulations complexes. Les enjeux sont en effet les mêmes : permettre à des thématiciens de comprendre et d'évaluer les sorties d'un modèle, à l'aide d'indicateurs nombreux et complexes présentés de manière transparente relativement à cette complexité.

Ce positionnement méthodologique s'inscrit pleinement dans la démarche de co-construction interdisciplinaire de SimFeodal. On y retrouve ainsi la même logique qui anime les dashboards : une évaluation menée par des thématiciens qui s'appuient pour cela sur des indicateurs clefs (ref. aux indicateurs les plus importants dans le chap 3) et précisent leur analyse à l'aide d'indicateurs secondaires présentés sous forme d'un panel varié de visualisations. Nous avons donc choisi de ré-organiser les rapports initialement produits pour leur donner une forme plus adaptée à ces enjeux, sous forme de *dashboards*.

5.2.4.2 SimVADB

Les *dashboards* font souvent usages de représentations graphiques très métaphoriques des tableaux de bords automobiles. On y retrouve fréquemment une forte mise en valeur d'indicateurs numériques simples au travers de représentations « skeuomorphes », c'est-à-dire qui reprennent l'apparence physique des objets symbolisés. On retrouve communément, par exemple, des indicateurs représentés sous forme de jauges (*gauge charts*), de thermomètres (*thermometer charts*), ou encore de voyants d'alerte et autres témoins lumineux (figure 5.4)).

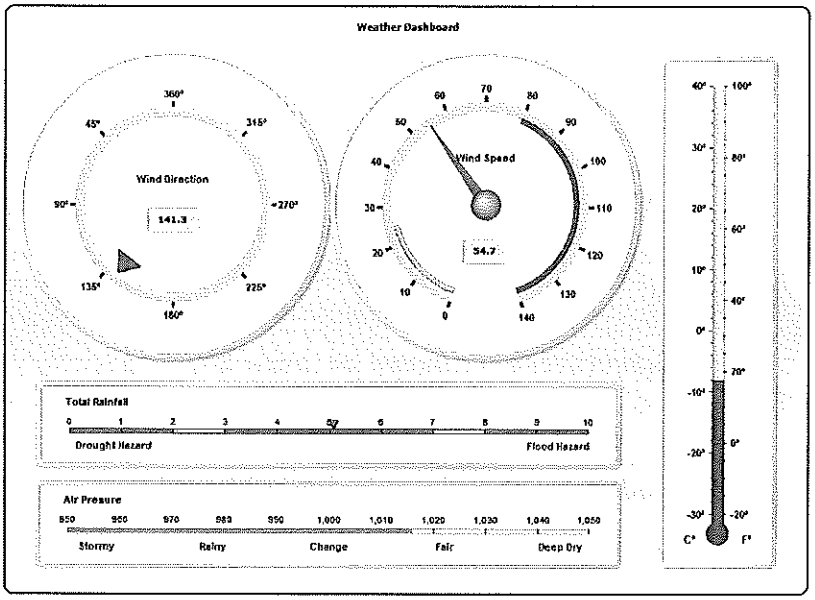


FIGURE 5.4 – Un exemple de représentations visuelles courantes dans les *dashboards*. Tiré de PANDRE (2011)

Pour SimFeodal, les indicateurs étant assez fortement conçus et structurés (chap. 3), nous n'avons pas ressenti le besoin de faire appel à ce type de représentation. Nous avons donc emprunté aux *dashboards* la logique d'organisation visuelle des indicateurs plutôt que les modes de visualisation en eux-mêmes. Pour faciliter la transition pour l'utilisateur, on cherchait ainsi à produire un *dashboard* au plus proche, visuellement, des rapports automatiques qui les précédaient.

On a pour cela développé un *dashboard* adapté à SimFeodal, nommé SimVADB [19]. Dans un premier temps, on souhaitait simplement ré-organiser le code-source produisant les rapports automatiques, afin de convertir ces rapports en *dashboards*. Cela a été effectué au moyen d'outils permettant de générer des applications en ligne, sans changer de langage de programmation. Dans ce cas, on s'est appuyé sur la librairie logicielle Flexdashboard (IANNONE, ALLAIRE et BORGES 2018).

Le passage du rapport automatique au *dashboard* illustre l'un des grands intérêts des outils de type *CLI* : dans le cas de SimVADB, il a suffit de réorganiser le code, sans modifier à aucun moment les fonctions de calcul et de création des indicateurs de sortie de simulation. Les codes 5.1 et 5.2 illustrent la facilité de cette modification. Il s'agit véritablement de placer les différentes fonctions dans des blocs graphiques. Ces modifications minimes augmentent toutefois considérablement la convivialité et la facilité de l'analyse de résultats de sortie d'un modèle.

19. Simulation Visual Analysis DashBoard.
Cette application a rapidement été remplacée par l'itération suivante (SimEDB, voir section 5.2.5), et n'a donc dans les faits jamais été complètement finalisée. On en trouve une trace, fonctionnelle mais incomplète (les versions ultérieures n'ont pas été enregistrées dans l'outil de versionnement), dans ce dépôt logiciel : https ://github.com/RCura/SimEDB/tree/2cd22c7c

```
# Agent de type A
  afficher('Agent de type A')

  print('Indicateur 1')
  calcul_indicateur_1 {...}
  affichage_indicateur_1 {...}

  afficher('Indicateur 2')
  calcul_indicateur_2 {...}
  affichage_indicateur_2 {...}
# Agent de type B
  afficher('Agent de type B')
  [...]
```

Code 5.1 – Pseudo-code du rapport automatique.

```
onglet{titre = 'Agent de type A',
  sous_onglet{titre = 'Indicateur 1',
    calcul_indicateur_1 {...}
    affichage_indicateur_1 {...}
  },
  sous_onglet{titre = 'Indicateur 2',
    calcul_indicateur_2 {...}
    affichage_indicateur_2 {...}
  }
},
onglet{titre = 'Agent de type B',
  [...]
}
```

Code 5.2 – Pseudo-code du dashboard.

Comme dans l'exemple de code, on a préféré organiser les indicateurs au seins d'onglets plutôt que de les présenter dans des pages successives. Les onglets de premier niveau représentaient les types d'agent, et des onglets de second niveau permettaient de visualiser l'ensemble des indicateurs associés à ces agents (figure 5.5).

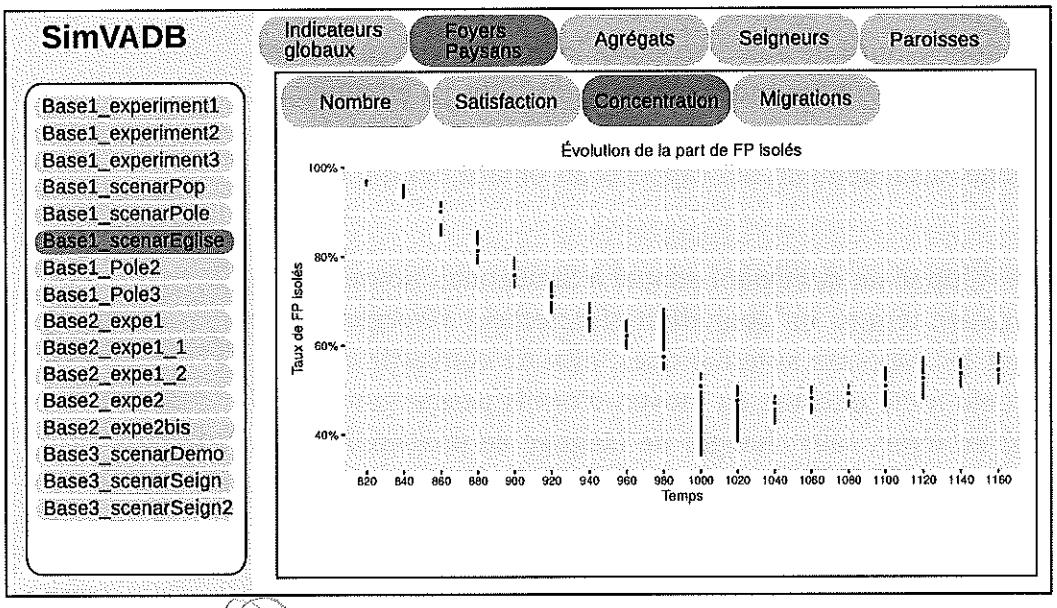


FIGURE 5.5 – Un *mock-up* de la première interface de SimVADB, un *dashboard* dédié à la visualisation des indicateurs de sorties de simulation de SimFeodal.

Au niveau de l'interface utilisateur, SimVADB permettait de choisir, via un menu de sélection (partie de gauche dans la figure 5.5), les expériences passées dont on voulait visualiser les indicateurs de sortie (les deux niveaux d'onglets de la partie de droite).

Les limites du *dashboard* Avec la multiplication des valeurs de paramètres testées, il est devenu plus efficace de regrouper les expériences au sein d'expérimentations. Celles-ci voient varier plusieurs paramètres, potentiellement avec de multiples valeurs de paramètres pour chacun. Elles constituent donc un ensemble d'expériences qui partagent des mécanismes et un jeu de

paramètre par défaut communs.

Avec le mode de sélection choisi dans SimVADB, basé sur le nom des expériences, il devenait plus difficile de sélectionner rapidement des ensembles d'expériences membres d'une même phase d'expérimentation. En effet, et comme illustré dans la figure 5.5, les noms d'expériences tendent à s'allonger, et avec leur masse augmentant, il est peu commode d'avoir à parcourir tout un long menu de sélection pour trouver les expériences souhaitées. De plus, malgré des tentatives de nommage régulières et explicites, la multiplication des expériences et expérimentations implique aussi une certaine confusion dans les types de mécanismes et valeurs de paramètres associés. Sans table de correspondance complète entre les noms des expériences et leurs valeurs de paramètres, il devenait impraticable de retrouver les différentes expériences mettant en avant, par exemple, des attractivités fortes par les pôles, une plus forte hiérarchisation des attracteurs ou encore des migrations lointaines facilitées. Le choix méthodologique d'interaction avec la plate-forme d'affichage des indicateurs, basé sur une sélection des expériences depuis leur nom, s'est donc révélé inadapté à la sélection et à l'exploration des sorties de SimFeodal.

Au delà du soucis du mode d'interaction, qui aurait pu être amélioré, un autre problème apparaissait. Pour évaluer visuellement différentes configurations du modèle, on ne pouvait se contenter d'un simple affichage des données, au sein d'un outil de type présentoir interactif tel que SimVADB. Comme dit dans le chapitre 4, le paramétrage de SimFeodal a ainsi reposé sur de nombreuses étapes d'évaluation des différentes version du modèle. L'approche d'analyse principale était donc la comparaison, point par point, entre les résultats des indicateurs de sortie de simulation des versions successives de SimFeodal. Un outil de présentation dynamique de résultats de sorties de simulations est certes plus adapté qu'un rapport statique, mais il ne constitue pas pour autant un outil adapté à la comparaison. S'il suffit pour de la restitution, par exemple dans le cadre du rapport systématique des résultats de SimFeodal, on ne peut s'appuyer uniquement sur une succession d'évaluations visuelles pour appréhender l'étendue des changements apportées par une modification des valeurs de paramètres.

5.2.5 Interagir avec les rapports : exploration interactive

Face à la démultiplication des expérimentations, consécutive aux nombreuses étapes de paramétrage de SimFeodal (cf. chap 4), il a fallu repenser la plate-forme d'évaluation des résultats. Pour cela, considérant que les simulations ne pouvaient être aisément appréhendées et sélectionnées par leur nom, numéro d'étape ou de version, il a été décidé d'adopter une posture plus proche de l'exploration du modèle en elle-même. C'est-à-dire de ne pas caractériser les simulations par un identifiant quelconque, mais plutôt par leur spécificité intrinsèque, c'est-à-dire la combinaison de valeurs de paramètres qui les rendent uniques. Ce faisant, au sein de la plate-forme d'exploration SimVADB, l'enjeu devenait plutôt la compréhension des effets des valeurs de paramètres sur les indicateurs que l'évaluation d'une simulation en particulier. Il fallait passer du

descriptif, quelle qu'en soit la méthode, à du comparatif.

Du point de vue de l'interface utilisateur, cela implique que la sélection ne se fasse plus par un unique critère (le nom de la simulation), mais au contraire sur du multi-critère. Par une succession de sélections, chaque paramètre pouvait constituer un nouveau filtre dans lequel on avait à choisir les valeurs à interroger (voir la figure 5.6 **(E)** de l'encadré 5.1).

La quantité de paramètres en entrée était importante et pouvait dès lors donner lieu à un mode de sélection complexe et fastidieux – définir une par une les valeurs voulues pour chacun des 45 paramètres –. Nous avons choisi encore une fois de nous appuyer sur l'aspect visuel afin de permettre aux utilisateurs de SimVADB de choisir la ou les expérimentations à analyser.

Visualiser avec des coordonnées parallèles Pour cela, on a choisi de représenter les combinaisons de paramètres dans un graphique en « coordonnées parallèles » (*parallel coordinates*, d'après INSELBERG et DIMSDALE 1987, voir FEW 2006b par exemple pour une description plus succincte, illustrée et pratique). Ce type de graphique est en effet extrêmement pertinent pour représenter une information multi-dimensionnelle en ce qu'il permet de détecter graphiquement des *clusters* d'individus statistiques[20] (HEINRICH et WEISKOPF 2013, p. 2), c'est-à-dire de faire ressortir visuellement les expériences dont les valeurs de paramètre sont proches. Notons bien que l'on parle ici des valeurs de paramètres, c'est-à-dire des conditions des expériences, et non des indicateurs de sortie. L'approche va ainsi des paramètres aux résultats : depuis des valeurs de paramètres choisies, on analyse la diversité des résultats.

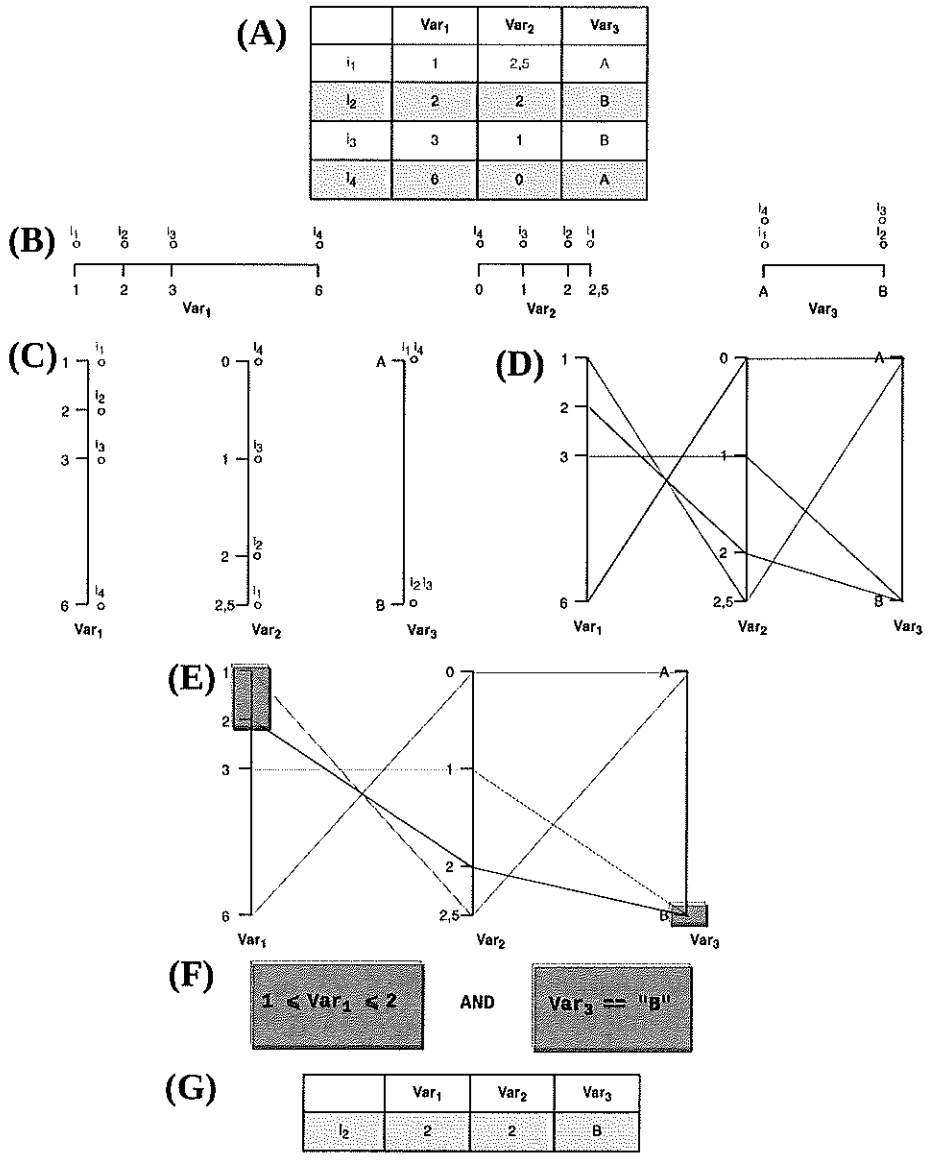
Interagir avec des coordonnées parallèles De plus, en matière d'interaction, on utilise fréquemment les graphiques en coordonnées parallèles en vue de filtrage. Cette opération est le plus souvent menée par des actions de *brushing* (« brossage »), c'est-à-dire de sélection graphique d'une zone en dessinant son étendue à la souris (voir encadré 5.1). Ce type de sélection se révèle en effet souvent plus efficace et intuitive qu'une sélection textuelle plus systématique :

> « Filtering is an operation that removes signals from its input. A filter reduces the number of lines to be rendered. In this sense, dynamic querying [...] is a filter, if implemented with brushing [...], which reduces clutter by putting the filtered lines in focus using some highlighting mechanism. Combining simple brushes using logical operators [...] further allows the user to formulate rather complex queries that might even achieve faster and more accurate results using parallel coordinates than using a Structured Query Language (SQL) [...]. »
>
> HEINRICH et WEISKOPF 2013, p. 13

20. Ici, chaque expérience est un individu statistique. Il est caractérisé par un ensemble de variables, les différents paramètres, et les modalités de ces variables que cet individu emprunte (les valeurs de paramètres). Quand le nombre d'individus est important, les différentes « courbes », qui correspondent au profil des individus sur le graphique en coordonnées parallèles, peuvent se superposer et montrer des tendances similaires. Avec ce type de représentation, il est facile de visualiser les grandes classes d'individus constituées par ces « tendances », et donc de constater des distinctions entre les individus (les expériences) de manière visuelle.

Encadré 5.1 : Construction et utilisation interactive d'un graphique en coordonnées parallèles

La figure 5.6 illustre les étapes successives de construction d'un graphique en coordonnées parallèles, depuis le tableau statistique (**A**) jusqu'au graphique final (**D**).
Pour cela, on projette les valeurs des variables sur des axes représentant chacune des variables (**B**). En normalisant la taille de ces axes et en les plaçant en parallèle (**C**), on peut alors tracer les « profils » des variables en reliant les positions de chacun des individus statistiques sur chacun des axes (**D**).
La seconde partie de la figure représente le mode d'interaction par *brushing* : on « brosse » sur chaque axe une sélection de valeurs à conserver (**E**). La sélection graphique est convertie en intervalles numériques et formalisée sous une forme classique (**F**) qui permet de filtrer les données sous-jacentes. Au final, cette opération renvoie le seul individu statistique répondant aux deux sélections graphiques (**G**).



	Var_1	Var_2	Var_3
i_1	1	2,5	A
i_2	2	2	B
i_3	3	1	B
i_4	6	0	A

$1 \leq Var_1 \leq 2$ AND $Var_3 == "B"$

	Var_1	Var_2	Var_3
i_2	2	2	B

FIGURE 5.6 – Construction d'un graphique en coordonnées parallèles et sélection interactive.

Cette utilisation est aussi courante en géographie quantitative, et on la retrouve par exemple chez l'un des représentants de l'analyse spatiale des années 1990, Stewart Fotheringham. Cet auteur indique même l'usage du graphique en coordonnées parallèles en tant que filtre pour identifier des informations dans une autre dimension, spatiale ici : « the data being displayed in parallel coordinates can be linked to a map and then brushed to highlight the locations of interesting lines displayed in *m*-space on the parallel co-ordinates. » (FOTHERINGHAM 1999).

Appliqué aux données de SimFeodal, cette interface (figure 5.7) se révèle particulièrement efficace pour sélectionner les configurations de paramètres à explorer. Ainsi, en « brossant » quelques filtres manuellement (figure 5.7 - **A**), on arrive rapidement à isoler une expérience spécifique.

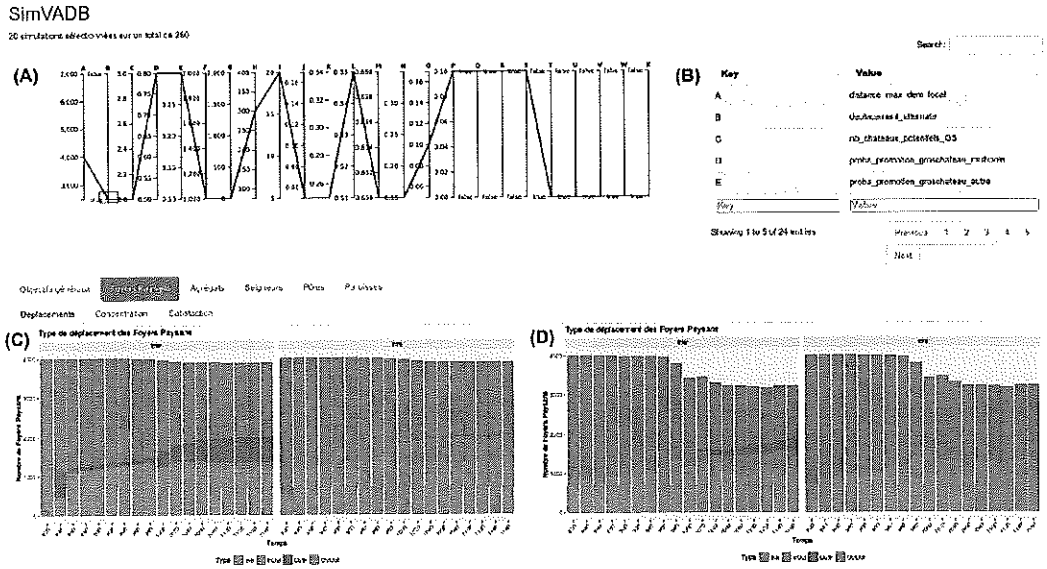


FIGURE 5.7 – SimVADB (Simulation Visual Analysis DashBoard), un *dashboard* d'exploration visuelle des indicateurs de sortie de simulation de SimFeodal.
La sélection des simulations à explorer se fait dans le graphique en coordonnées parallèles (**A**), en « brossant » des filtres graphiques sur les « dimensions » du graphique, dimensions dont les intitulés sont explicités dans le tableau **B**. Les graphiques **C** et **D** indiquent l'évolution des types de déplacements des foyers paysans au cours des simulations.
- Le graphique **C** représente, pour cet indicateur, une moyenne de l'ensemble des simulations intégrées dans la base de données (260 ici), recouvrant donc plusieurs valeurs de paramètres.
- Le graphique **D** représente cet indicateur calculé depuis un sous-ensemble de 20 simulations (donc une expérience composée de 20 réplications), pour lesquelles le paramètre « B » (deplacement_alternate) vaut true.

Afin de permettre aux utilisateurs de remarquer les particularités des simulations explorées, nous avons choisi de mettre en emphase les différences entre la tendance générale des indicateurs, en calculant des moyennes de l'ensemble des simulations (figure 5.7 - **C**), et les valeurs spécifiques des indicateurs de l'expérience choisie (figure 5.7 - **D**). Cela permet, visuellement, d'être en mesure d'évaluer les sorties de simulation d'une expérience tout en ayant un référentiel visible. Les différentes expériences produisent des résultats sensible-

ment similaires [21], et on ne peut alors plus les comprendre sans les confronter à d'autres résultats similaires. Le choix d'une agrégation de l'ensemble des simulations effectuées est discutable, en ce qu'on aurait par exemple pu plutôt isoler des simulations « de référence » afin de diminuer l'effet « d'aplatissement » engendré par l'agrégation de résultats nombreux et hétérogènes. Toutefois, la variabilité des résultats étant encore assez restreinte, au moment de la création et de l'utilisation de SimVADB, ce référentiel agrégé permettait déjà une compréhension plus fine des sorties de simulations, en particulier dans l'analyse de l'impact de variations fines de valeurs de paramètres.

5.2.6 Explorer en comparant : SimEDB

Après un travail de paramétrage grossier qui permet de stabiliser les mécanismes, il est souvent nécessaire de passer à une phase plus fine. On vise à ce moment à mieux calibrer un modèle à l'aide de variations de valeurs de paramètres de granularité inférieures. En vue d'évaluer les simulations, et donc de les différencier les unes des autres à l'aune des indicateurs générés, la comparaison d'une expérience spécifique avec un référentiel constitué de toutes les expériences précédentes ne permet plus de mener ce travail de comparaison fine. Les variations entre simulations sont trop fines pour être distinguables les unes des autres par le biais d'une comparaison avec un référentiel unique [22]. Cela s'entend quelque soit la manière dont ce référentiel est constitué, qu'il résulte d'une agrégation de simulations ou encore d'une version « de base » du modèle (par exemple, dans le cas de SimFeodal, les versions principales identifiées dans le chapitre 4).

Pour pouvoir correctement évaluer les apports d'un nouveau jeu de valeurs de paramètres, et donc, dans une démarche itérative, pouvoir différencier deux expériences successives, il est nécessaire d'être en mesure de comparer directement les expériences les unes avec les autres, ou encore avec un référentiel facilement ajustable. On peut énumérer les quelques exemples de cas de figure suivants :

— Comparaison entre une expérience spécifique et une autre expérience spécifique de même « importance ». Par exemple, comparer deux expériences qui font varier légèrement différement une valeur de paramètre.

— Comparaison entre une expérience spécifique et une autre expérience

21. Si chaque expérience, et chaque réplication, produisent des résultats uniques, le choix d'une évaluation par des indicateurs visuels peut prêter à confusion si l'on n'a pas de repère précis. Les critères attendus, présentés dans le chapitre 3 sont ainsi assez précis pour départager une simulation très éloignée des attentes et une autre simulation plus conforme. Pour autant, par exemple quand les valeurs de paramètres varient faiblement, les résultats produits peuvent être assez similaires dans les grandes tendances qu'ils font ressortir.

22. On peut prendre l'exemple de SimFeodal. En faisant varier le nombre de foyers paysans de 4000 à 4200 (5% de variation), les résultats du modèle changent peu : de faibles variations de valeurs de paramètres entraînent le plus souvent de faibles variations dans les indicateurs de sorties observés. On peut de plus noter que les répercussions d'un changement de valeur de paramètre peuvent être très différentes de l'ordre de grandeur de ce changement de valeur. Cela s'explique par la non-linéarité de l'influence des paramètres. Dans l'exemple pris, pour ces 5% de variation dans le nombre de foyers paysans, la majorité des indicateurs variera ainsi de moins de 1%.

spécifique d'« importance » différente. Cette différence de niveau peut être constituée par exemple par une version « de base » que l'on comparerait à une variante de celle-ci.

— Comparaison entre deux expérimentations, par l'agrégation de leurs résultats. Si l'on a mené des expérimentations faisant varier de manière systématique deux paramètres différents, il peut être intéressant de les comparer en bloc, c'est-à-dire par exemple en prenant les moyennes de chacune des expériences composant ces expérimentations.

Ce faisant, on fait en fait varier la notion de « simulation de référence », qui peut alors revêtir plusieurs formes. Pour cela, il n'est plus possible de mener une comparaison visuelle entre un référentiel commun et une expérience spécifique, mais bien de baser l'évaluation sur la comparaison entre deux ensembles spécifiques qui doivent pouvoir être spécifiés. D'un point de vue méthodologique, cela requiert de pouvoir afficher conjointement les indicateurs de sorties de deux expériences (ou ensembles d'expériences). Cela implique aussi de laisser à l'utilisateur la responsabilité d'un choix supplémentaire puisqu'il faut désormais effectuer deux sélections : une pour chacun des points de comparaison. La sélection d'une expérience via l'usage de *brushing* sur un graphique en coordonnées parallèles des valeurs de paramètres ayant montré son efficacité, il a été choisi d'étendre ce principe d'interactivité au choix du référentiel.

Dans cette version remaniée de la plate-forme d'exploration (voir figure 5.8), renommée SimEDB (**Sim**Feodal **E**xploration **D**ash**B**oard) [23], l'accent est mis sur la comparaison de deux ensembles de résultats, chacun répondant à une sélection propre. L'utilisateur doit ainsi « paramétrer » interactivement, via *brushing*, les expériences à afficher pour le référentiel et pour les expériences à comparer. On dispose pour cela de deux outils de filtrage des simulations (partie de gauche dans la figure 5.8), qui peuvent être utilisés de concert, pour comparaison visuelle, ou par étapes successives [24].

En superposant les graphiques et tableaux des indicateurs, la comparaison visuelle est facilitée. On peut alors comparer deux variations fines d'un mécanisme du modèle, en sélectionnant par exemple une unique différence dans les valeurs de paramètres du modèle (par exemple un paramètre relatif à la promotion des paroisses dans la figure 5.8). De manière générale, ce choix d'outil d'interrogation des données permet de répondre à l'ensemble des cas de figures identifiés dans les paragraphes précédents.

23. La plate-forme d'exploration SimVADB (SimFeodal Visual Analysis Dashboard) a été renommée SimEDB ([...] Exploration Dashboard) par soucis de simplicité, le terme « Exploration » nous semblant plus compréhensible et explicite que celui de Visual Analysis. Ce nom apporte de plus une cohérence sémantique entre plusieurs productions de l'auteur – TimeLineEDB - (CURA 2017b); RoadTrafficEDB et CitiBikeEDB - (CURA 2017a). Cela inscrit cette plate-forme d'exploration de données issues de simulation dans une « famille » d'outils d'exploration de données spatio-temporelles.

24. En menant par exemple une première comparaison entre une expérience « A » en haut et « B » en bas, puis en sélectionnant « C » en haut, puis « D » en bas etc. On compare ainsi A avec B, puis B avec C, et enfin C avec D.

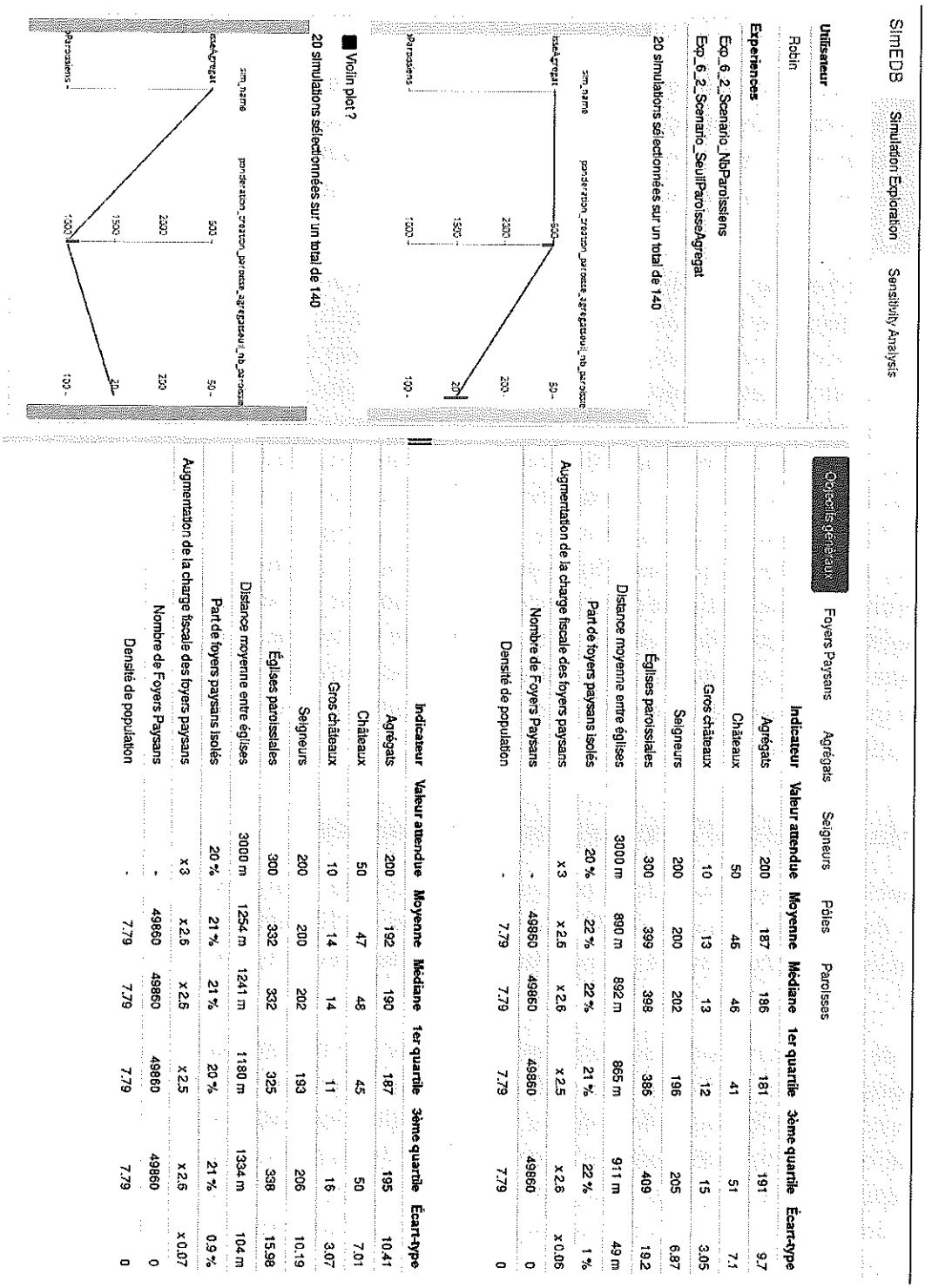


FIGURE 5.8 – SimEDB

Nous reviendrons plus précisément et longuement sur la description de SimEDB dans les parties suivantes (section 5.4, p. 56), mais après en avoir décrit les étapes de construction et les besoins auxquelles ces évolutions répondaient, il est maintenant nécessaire de revenir sur les données manipulées par cette plate-forme d'exploration. Le type, la structure et la masse de ces données (section 5.1) sont en effet indissociables des choix méthodologiques effectués pour SimEDB. Il est donc important de présenter les choix et contraintes de ces données avant d'entrer dans une description approfondie de la plate-forme.

5.3 Organiser les données

On a vu dans la première partie de ce chapitre (section 5.1) que les données produites par SimFeodal étaient nombreuses, diverses et massives. La seconde partie (section 5.2) a montré les types de problèmes que de telles données, une fois exploitées pour en tirer les indicateurs de sortie, peuvent poser en matière d'exploration.

Nous souhaitons ici revenir sur une corollaire indispensable à l'utilisation efficace de ces données. De la même manière que la multiplicité des indicateurs, des expériences et des expérimentations requiert des outils d'exploration adaptés, la multiplicité des données requiert des outils de stockage et d'interrogation eux aussi adaptés. Là encore, on peut noter une succession de contraintes liées à ces données et à leur massification, contraintes qui limitent et guident les modes d'organisation de ces données. Sans structuration adéquate, l'acquisition, l'archivage, l'interrogation ou encore la sauvegarde des données générées par le modèle ne peuvent être garantis, et encore moins de manière efficace. Le choix d'une méthode d'organisation des données en sortie de simulations ne relève donc pas d'une quelconque coquetterie technique. Au contraire, conditionne et contraint fortement aussi bien les modalités de création des indicateurs que les possibilités de la plate-forme d'exploration de les afficher de manière interactive.

5.3.1 Assurer la capacité d'interrogation des données

Avant de se soucier du « schéma » de la base de données[25], du choix du Système de Gestion de Base de Données (SGBD), ou encore des performances de ce dernier, il convient de se fixer sur la manière dont on souhaite entreposer les données.

De la myriade de fichiers issus de tableurs organisés dans une multitude de dossiers spécifiques à l'entrepôt de données décentralisé orienté documents, en passant par les traditionnelles bases de données relationnelles, les possibilités de stockage et d'organisation des données issues de simulations sont ainsi innombrables. Plusieurs contraintes successives permettent de limiter le choix à un sous-ensemble de solution adaptées parmi lesquelles on peut alors mener une étude plus approfondie, passant notamment par des comparaisons entre ces candidats.

Une des premières contraintes est constituée par la nécessité d'interroger fréquemment et de manière répétée les données. C'est l'une des contreparties du passage d'un rapport automatique à un outil d'exploration interactif (section 5.2.4). Ainsi, dans un rapport automatique, on interroge une fois les données pour en tirer les indicateurs de sortie, et ceux-ci ne sont plus amenés à changer, sauf re-calcul par exemple suite à des ajouts d'indicateurs. Au

25. On utilise souvent le terme de « schéma » pour désigner la version implémentée, dans un SGBD spécifique, du Modèle Conceptuel de Données (MCD). Contrairement au MCD, qui donne une version conceptuelle et générique d'une base de données, le schéma est donc tributaire du SGBD dans lequel il est intégré.

contraire, dans un outil interactif, les indicateurs sont calculés depuis les données « à la volée », c'est-à-dire à chaque qu'un indicateur doit être affiché. On peut mettre en place un système de cache, pour conserver les calculs déjà effectués, mais avec les dernières itérations de SimEDB où le nombre de possibilités de sélections est extrêmement important, ce n'est plus possible. Il est nécessaire de procéder aux calculs à chaque affichage des graphiques et tableaux de résultats. L'interrogation des données est donc extrêmement fréquente et répétitive. Elle doit alors être aussi simple (en termes de mode d'interrogation) qu'efficace (en termes de rapidité d'interrogation).

Dans le cadre des données issues de SimFeodal, en vue de leur mobilisation dans SimEDB, nous avons ainsi eu à sélectionner quelques SGBD candidats parmi une foule de solution possibles. Afin de guider ce choix, trois critères ont été définies, et forment, selon une hiérarchie propre à leur ordre, un ensemble de filtre ayant permis la réduction des SGBD possibles à un nombre appréhendable. Ces critères, dont l'énonciation guidera cette partie, sont ainsi (1) l'universalité, ou « agnosticité », des SGBD aux outils de requête; (2) la pérennité et stabilité des solutions disponibles et (3) les performances des SGBD considérés.

5.3.1.1 Interroger de manière universelle et indépendante

Lors de la conception d'un outil faisant appel à des données, qui plus est massives, il convient de se positionner tôt sur la manière d'interroger ces données. Par interrogation, on entend ici, comme souvent dans le domaine des bases de données, la manière de faire appel, concrètement, aux données, pour en tirer les sous-ensembles, agrégations et autres résultats synthétiques résultant du traitement des données brutes. Si l'on considère des données stockées dans un tableur, alors les « formules », les tableaux croisés dynamiques ou encore les graphiques issus du tableur sont des interrogations des données, qui s'expriment dans ce cas via un ensemble de langages, écrits – les formules, qui font appel à des fonctions spécifiques des tableurs – ou visuels – les tableaux croisés dynamiques, construits en faisant glisser des intitulés de colonnes dans un tableau.

Stockage distribué ou centralisé Avant même de s'intéresser aux spécificités de ces langages, un premier choix réside dans le mode de stockage des données qui doivent être mis à disposition d'une plate-forme. Doit-on laisser l'utilisateur intégrer lui-même les données, et ainsi, en faire un stockage « distribué », dans le sens où chaque utilisateur de l'application possèderait physiquement une copie locale des données? Ou, au contraire, les données doivent-elles être centralisées, c'est-à-dire enregistrées en une seule copie à laquelle les utilisateurs accèderaient à distance? Pour reprendre l'exemple des tableurs, doit-on privilégier une solution locale – chacun ayant une copie du fichier tableur et menant ses propres modifications dessus – ou une approche de type centralisée, par exemple en privilégiant des tableurs collaboratifs en ligne (*Google Docs/Sheets* par exemple)?

Habituellement, c'est-à-dire dans une grande majorité d'applications, les

données sont stockées localement : cela permet, en particulier, de ne pas dépendre d'une connexion internet pour interroger des données qui seraient hébergées sur internet. Dans le cas de SimFeodal, cette solution est rendue difficile, sinon impossible, par la masse de données en sortie de simulations. Si chaque utilisateur de SimEDB devait posséder une copie des données, voir plusieurs en cas d'utilisations depuis différents ordinateurs, cela occuperait plusieurs gigaoctets de données à chaque fois. De plus, en cas de mise à jour des données, c'est-à-dire d'insertions de nouvelles sorties de simulations, il faudrait distribuer à nouveau l'ensemble du jeu de données, à chaque fois.

Pour ces raisons, nous avons fait le choix d'un stockage centralisé, sous forme d'une architecture « client-serveur », hébergé sur un serveur internet dédié, ce qui permet à la plate-forme de travailler à chaque fois sur les données les plus à jour, réduit la taille du stockage physique associé, et dispense d'une configuration sur chaque poste utilisateur : si le lien entre l'application et les données fonctionne correctement pour un utilisateur, il fonctionnera à l'identique pour tous les autres. Ce choix présente un dernier avantage, non négligeable : en stockant les données en un seul lieu, c'est-à-dire sur un serveur informatique, on peut faire en sorte de rendre ce serveur aussi performant que possible, et accélérer ainsi l'interrogation des données pour tous les utilisateurs.

Interrogation spécifique ou générique De nombreuses solutions intégrées de gestion de données proposent leurs propres modes d'interaction avec les données, c'est-à-dire un langage spécifique permettant d'interroger les données contenues dans le système[26]. Au contraire, les SGBD les plus classiques s'appuient plutôt sur des langages de requêtes aussi standardisés que possibles, afin de faciliter l'adoption de leur propre solution à des utilisateurs d'autres plate-formes. La spécificité présente l'avantage de langages plus adaptés aux données manipulées, et donc souvent plus intuitifs dans l'interrogation des spécificités des données. De plus, la spécificité permet aussi une optimisation des requêtes, et est donc souvent plus performante que les solutions plus génériques.

De manière générale et dans le cas de SimFeodal, nous avons préféré privilégier une approche plus générique, faisant appel à des solutions de SGBD plus standardisées. La raison tient principalement à une volonté de généricité du stockage des données : au cours des différentes étapes de construction de SimFeodal, les besoins en matières d'interrogation des données ont évolué. Cette évolution était prévisible et prévue, et nous avons donc choisi dès le départ d'adopter uniquement des solutions modulaires, garantissant une évolutivité facilitée de la base de données, aussi bien regardant sa structure (enregistrement de nouvelles variables ou de nouveaux agents du modèle) que son contenu (massification des données en sortie au fur et à mesure de l'exploration du modèle).

26. Souvent, cette interrogation se fait par appel à des *API* (*Application Programming Interface*, ou Interface de Programmation Applicative en français). Ces interfaces sont propres à chacune des plate-formes, et demandent donc un langage et un formalisme de requête spécifique.

De plus, dans la perspective de ce travail de thèse, où l'on cherche à rendre les productions aussi reproductibles et génériques que possible, il était indispensable de disposer d'un SGBD aussi standard que possible pour en faciliter l'adoption et l'adaptation à d'autres modèles de simulations par exemple.

Bases de données relationnelles ou NoSQL Même une fois arrêté sur le choix de ne faire appel qu'à des outils standards pour stocker les données, le nombre de solutions disponibles demeure très important. Afin de réduire ce nombre, on peut déjà choisir les grands types de SGBD auxquels faire appel. Les SGBD sont ainsi souvent classés selon les grands traits de la méthode dont ils organisent les données. Les deux grands types[27] sont les SGBD « relationnels » et les SGBD « NoSQL ». Si la distinction est sujette à de très nombreux débats, souvent virulents[28] , on se contentera ici de définir les SGBD relationnels comme les SGBD, les plus fréquemment utilisés, où l'information est stockée dans des tables composées de champs – les colonnes, correspondant aux variables – et de lignes – les entités décrites par les variables. Le format des données est donc rectangulaire et n'accepte pas, comme dans un tableau statistique, que les entités possèdent un nombre de variables différent, ou encore des types de valeurs différents de celles des autres entités (une même colonne ne peut donc contenir conjointement un nombre et un texte dans des lignes différentes). On nomme ces bases de données relationnelles via la manière qu'elles ont de faire communiquer des éléments hétérogènes, et donc contenus dans des tables différentes : si les tables ont une colonne en commun, on pourra alors effectuer une opération de jointure permettant de mettre en commun les informations de ces tables dans une unique table résultante.

A l'inverse, les SGBD NoSQL se définissent de manière opposée à ce mode de stockage[29], rompant par exemple la contrainte d'unicité de type des colonnes, ou de nombre identique de colonnes renseignées pour chaque entité. Pour simplifier le discours, on se contentera de caractériser les SGBD NoSQL comme des SGBD non relationnels. Les SGBD NoSQL ont, en général, de bien meilleures performances et une plus grande flexibilité que les SGBD relationnels. Dans le cas de SimEDB/SimFeodal, où l'on est confronté à des données massives, cela présente un avantage non négligeable.

Toutefois, leur flexibilité est associée à une contrainte majeure en termes de généricité : alors que les SGBD relationnels partagent un langage d'interro-

27. Il en existe d'autres, comme les SGBD orientés objets (quasiment disparus aujourd'hui), orientés graphes (Neo4j...), les SGBD pensés pour le stockage et l'interrogation d'ontologies (*Triplestores RDF*, interrogeables en langage SPARQL) ou encore les nouveaux SGBD de type « NewSQL » (Apache Ignite, CockRoachDB...) pensés pour une parallélisation massive des données. Ces types de SGBD ne correspondent toutefois pas du tout aux besoins identifiés pour SimEDB/SimFeodal, et sont en général dédiés à des problèmes et marchés de niches. Nous ne les décrirons donc pas plus en détail ici.

28. À l'instar des violentes querelles qui agitent régulièrement les informaticiens : Vim vs Emacs, Programmation Orientée-Objet vs Programmation Fonctionnelle, R vs Python...

29. À l'origine, c'était le sens fort du nom « NoSQL » : Non SQL, le SQL faisant ici références aux SGBD pré-existants, majoritairement relationnels, dont la mouvance NoSQL, portée par l'apparition des « *big data* » a voulu se distinguer. Sans entrer dans le détail, notons tout de mêmes que de nombreux SGBD NoSQL, qui traduisent désormais cet acronyme par « Not only SQL », sont maintenant relationnels, mais mettent en avant d'autres types d'approches.

gation commun, le SQL (*Structured Query Language*)[30], les SGBD NoSQL font plus souvent appels à des langages spécifiques à chaque SGBD. Pour SimEDB, cela impliquerait une fort dépendance au SGBD choisi : en cas de changement de SGBD, toutes les requêtes seraient à reformuler dans le nouveau langage, parfois même selon des logiques extrêmement différentes les unes des autres (dérivés du SQL, interrogations via des objets JSON, via des langages de parcours de graphes...) etc. Au contraire, avec les SGBD relationnels, le langage de requête étant commun, une fois le code d'interrogation généré, il est très aisé de changer de SGBD cible. Cela garantie une forte capacité d'évolutivité aux outils d'interrogation de données tels que SimEDB. Puisque les fournisseurs de données sont interchangeables, on peut en changer au fur et à mesure de l'apparition de nouveaux besoins.

En raison de la généricité de ces solutions relationnelles, qui vient s'ancrer dans la recherche de reproductibilité et de généricité de notre démarche d'ensemble, nous avons donc choisi de faire reposer le stockage et l'interrogation des données sur des SGBD relationnels. Cette décision s'est montrée d'autant plus heureuse que, au cours de la construction et de l'évolution de SimEDB, le SGBD choisi pour héberger les données a changé plusieurs fois. La généricité des outils choisis a permis de minimiser, voire quasiment d'éviter, les changements à apporter au code-source de SimEDB relatif à l'interrogation de données en vue de produire les indicateurs de sortie depuis les données brutes en sortie de SimFeodal.

Entrepôts de données et interrogation directe En parallèle des SGBD, des solutions « intermédiaires » permettent de s'abstraire des SGBD en eux-mêmes pour mener les requêtes. Ces solutions, que l'on nomme « Entrepôts de Données » (*Data Warehouses*), se comportent comme une surcouche faisant l'interface entre un ou plusieurs SGBD et la requête émise par le client final. Elles se placent donc comme intermédiaire entre les SGBD mobilisés et les applications qui les interrogent. Les entrepôts de données jouent aussi bien le rôle d'agrégateurs de données[31] que d'environnements de manipulation et de restructuration de données (on les nomme alors « ETL » – *Extract-Transform-Load*).

Le grand intérêt de ces outils d'interface est d'abstraire la complexité de chacune des bases de données manipulées en générant une interface d'interrogation unique et générique, souvent performante grâce à des optimisations spécifiques (pré-calcul des requêtes possibles par exemple).

Dans les paragraphes suivants, Lena note qu'elle a perdu le fil mais que c'est bien écrit.

30. Ce langage d'interrogation est omniprésent dans l'interaction avec les SGBD, mais aussi, avec de légères variantes, au sein de nombreux logiciels reposant des sélections de données, par exemple les logiciels SIG qui se basent sur la syntaxe du SQL (le fameux triptyque `SELECT ... FROM ... WHERE ...`)

31. C'est-à-dire qu'ils permettent d'agréger des sources de données composites, provenant potentiellement de différentes sources (plusieurs bases de données relationnelles) et de différents types de sources (différents SGBD, relationnels ou non par exemple).

Dans le domaine de la visualisation interactive de données, ces outils sont beaucoup utilisés, en particulier dans le monde de l'informatique décisionnelle. Ils se révèlent en effet extrêmement utiles quand les données sources ne peuvent être modifiées (par exemple quand elles sont issues de chaînes de collecte complexes, ou encore quand leur volumétrie et leur débit est important), puisqu'ils permettent de constituer une surcouche rendant l'interrogation et la visualisation de ces données accessibles à des analystes non spécialistes de la manipulation de données. Toujours en informatique décisionnelle, il est courant de faire appel à des entrepôts de données d'un type spécifique, les « traitements analytiques en lignes », ou environnements OLAP (*OnLine Analytical Processing*), qui permettent de structurer, par exemple sous formes de cubes de données, des sources de données hétérogènes présentant de nombreuses dimensions.

Les environnements OLAP ont été utilisés, promus et adoptés dans le champs scientifique de la géomatique, en ce qu'ils permettent de mettre en place rapidement des environnements d'analyse visuelle de données multi-dimensionnelles spatiales et temporelles. Dans ce cadre, où ces outils sont appelés « SOLAP » (*Spatial OLAP*), les données spatiales s'intègrent extrêmement bien en raison de leur capacité à s'emboîter selon les échelles, ouvrant dès lors la voie à des analyses multi-échelles et multi-dimensionnelles complexes.

Dans la communauté géomatique francophone, les solutions SOLAP sont bien représentées (par exemple autour de Sandro Bimonte et de son travail de visualisation de données spatiales environnementales, (BIMONTE 2007 ; BIMONTE, TCHOUNIKINE et MIQUEL 2005 ; ZAAMOUNE et al. 2013), et sont couramment employées pour répondre à des questionnements méthodologiques proches de ceux développés dans cet ouvrage. En lien avec les besoins de performances identifiés plus haut, on notera que certaines solutions OLAP permettent aussi d'optimiser la vitesse d'interrogation de bases de données, et visent ainsi à garantir une réponse rapide pour des outils d'interrogation de données interactifs (ZENG, AGARWAL et STOICA 2016).

Nous avons cependant choisi de ne pas faire usage de ces outils pour les mêmes raisons que pour les SGBD NoSQL : les avantages qu'ils présentent ne suffisent pas à contre-balancer les défauts en termes de généricité qu'ils introduisent. Pour profiter au mieux de ces environnements, il est en effet nécessaire de faire appel à un nouveau langage d'interrogation des données (le « MDX », de « *Multidimensional Expressions* »). Les différentes solutions OLAP/-SOLAP, de plus, présentent les mêmes inconvénients que les SGBD NoSQL : chacune interagit de manière propre aux différents SGBD, et ces outils sont donc difficilement interchangeables.

De la même manière, on se restreindra, parmi les SGBD relationnels interrogeables en SQL, à ceux qui disposent d'une méthode d'interrogation standard : si tous ces SGBD acceptent le SQL, certains demandent par exemple des protocoles spécifiques pour recevoir établir la connexion au SGBD, recevoir la requête et renvoyer les données correspondantes. Pour ce même choix « d'agnosticité » de la plate-forme d'interrogation face à la solution de stockage

choisie, on ne conservera que les SGBD acceptant les connexion standardisées (ODBC et JDBC).

SGBD et données spatiales On a mentionné le fait que les entrepôts de données étaient fortement utilisés, en particulier dans la communauté géomatique, car très appropriés aux données spatiales. De prime abord, ce point peut paraître critique : jusque là, on s'est contenté de mentionner les capacités organisationnelles de SGBD, et non leur aptitude à manipuler des données spatiales. Ce point, dans les SGBD relationnels, constitue un filtre important : sur la centaine de solutions disponibles, seule une poignée est en mesure de stocker efficacement et d'interroger de l'information spatiale[32].

Pourtant, au regard des indicateurs de sortie de simulation sur lesquels on s'appuie pour évaluer le comportement de SimFeodal, une grande majorité est non spatiale, en raison de la difficulté à agréger des données spatiales théoriques. Dans SimFeodal, la plupart des indicateurs de sortie sont non spatiaux, c'est-à-dire qu'ils mobilisent plus la dimension attributaire des données que leur dimension géographique[33].

La gestion de données spatiales ne constitue donc pas une absolue nécessité, contrairement aux points évoqués auparavant. Elle peut toutefois se révéler avantageuse, ne serait-ce que pour permettre l'observation des configurations spatiales simulées. Cela constitue une approche idiographique, visant à exemplifier plus qu'à synthétiser, mais permet tout de même une compréhension rapide des changements de structures spatiales. Toutes choses égales par ailleurs, on privilégiera donc des solutions de stockage ayant une capacité à gérer les données spatiales.

Pour héberger et organiser les données produites par SimFeodal, en vue de leur interrogation dans SimEDB, nous avons choisi de restreindre la myriade de solutions disponibles grâce à plusieurs filtres successifs. En premier lieu, on a choisit de faire appel à des solutions centralisées (), au sein de Systèmes de Gestion de Base de Données (SGBD). Ces SGBD permettent une interrogation standardisée () via un langage de requête universel, le SQL (). On a ensuite décidé d'interroger ces SGBD sans passer par l'intermédiaire d'entrepôts de données, et au travers de connexion aussi standardisées que possible (). Les SGBD répondant à ces critères sont les SGBD « relationnels », dont certains possèdent qui plus est une capacité intéressante à stocker et interroger des données spatiales (), ce qui constitue notre dernier filtre.

32. Les données spatiales peuvent être stockées dans tous les SGBD si l'on attribue une représentation textuelle, en chaînes de caractères, par exemple en utilisant le format *Well-Known Text* (WKT). Pour autant, ce format est lourd, inadapté à une indexation, et ne peut permettre à un SGBD de mener des requêtes spatiales directement depuis ces entités. Il est ainsi, par exemple, impossible de calculer le centroïde d'un polygone directement depuis une représentation WKT, alors que c'est aisé avec un stockage géométrique.

33. Les raisons en sont multiples et on y reviendra largement dans le chapitre 7. Notons tout de même que les indicateurs résultent de l'agrégation des réplications, et que cette agrégation est extrêmement complexe sinon impossible sur des données spatiales majoritairement aléatoires (voir chapitre 2, section 2.2.2.1).

5.3.1.2 Interroger de manière robuste et efficace

En dépit de l'accumulation de critères exposée précédemment, une quantité importante de SGBD demeurent en lice. Afin de les différencier, nous avons choisi d'ajouter des critères qui ne portent plus sur les grands types de SGBD, mais plutôt sur une différenciation des SGBD relationnels existants. Ces deux critères, précisés par la suite, sont d'une part la robustesse des SGBD, et d'autre part leurs performances. Ces critères sont pas des « prétextes » à une hiérarchisation quantifiée des SGBD, mais ont une importance prépondérante dans notre cas d'utilisation.

Robustesse des SGBD Le premier critère ajouté est celui de la robustesse, c'est-à-dire, ici, de la capacité du SGBD à être interrogé de manière (1) stable et (2) pérenne dans le temps. Une même requête sur les mêmes données doit systématiquement renvoyer le même résultat (stabilité), quelle que soit la durée séparant ces requêtes (pérennité). Si la base de données n'est plus interrogeable quelques mois après sa configuration, ou qu'elle renvoie des résultats différents, alors elle ne peut constituer une solution crédible à l'exploration d'un modèle sur une période longue.

La **stabilité** des bases de données est principalement due à la manière de stocker l'information d'un point de vue informatique.
En premier lieu, l'information peut être contenue « en clair » ou alors de manière archivée. Un stockage « en clair » est plus facilement accessible, puisqu'on peut le consulter avec n'importe quel éditeur de texte. Un stockage archivé est moins universel, mais occupe généralement un espace disque inférieur et comporte des mécanismes de vérification de la cohérence des données. Il est donc plus stable.
Une second différenciation tient à l'emplacement lieu du stockage. Celui-ci peut être effectué dans un unique fichier, ce qui a l'immense avantage de la portabilité des données : pour faire migrer ou sauvegarder la base de données, il suffit de copier le fichier. La plupart des SGBD adoptent toutefois un mode d'organisation en plusieurs fichiers, notamment pour des questions de redondance et de vérification de l'intégrité des données : en multipliant les fichiers, on minimise le risque d'erreur critique sur l'ensemble des fichiers à la fois. On notera enfin un dernier type de SGBD, où l'information n'est pas stockée sur un disque dur, mais est entièrement contenue dans la mémoire vive de l'ordinateur : les SGBD « *in-memory* ». Ces SGBD sont les plus rapides et stables, mais il faut les re-constituer à chaque redémarrage du serveur qui les héberge, ce qui peut prendre un temps important.

L'enjeu du choix est de se prémunir de « corruptions » de la base de données : quand le SGBD ne comporte que pas ou peu de mécanismes de vérification de l'intégrité ou de la cohérence des données, il peut arriver qu'une base de données se corrompe. On peut prendre l'exemple de l'exécution d'une requête demandant un calcul complexe et long. Cette requête pourrait être interrompue en cours d'exécution par faute d'un *bug* ou d'une expiration de sessions (*timeout*). Dans ce cas, il se peut que la base de données s'arrête dans un état muté – avec une nouvelle table ajoutée pour moitié par exemple – et ne

soit donc plus intègre. C'est très fréquent pour les SGBD basés sur un unique fichier, ou encore stockés en clair, puisque les nouvelles informations de la base de données y sont ajoutées au fur et à mesure, plutôt que d'être intégrées dans un fichier annexe que l'on pourrait réinitialiser en cas d'erreur.
Avec la volumétrie des données produites par SimFeodal, les requêtes peuvent s'avérer très longues, et une erreur dans une requête peut fréquemment corrompre la base de données. En termes de stabilité, on se tournera donc plutôt vers des SGBD relationnels stables, basés sur une redondance des données et donc sur des architectures archivées et multi-fichiers.

La **pérennité** des SGBD est un sujet proche, tenant aussi à la capacité à interroger les données contenues dans une base de données, mais cette fois-ci du point de vue de l'interrogation en elle-même plutôt de des données sur lesquelles elle s'applique : si le SQL est un langage standard[34], les types de données intégrées varient cependant d'un SGBD à un autre (champs textuels ou d'entiers « courts » par exemple). SQL étant un langage typé, selon la manière (bas niveau) dont sont intégrées les données, certaines requêtes identiques peuvent renvoyer des résultats différents selon les SGBD. Plus gênant, les normes implémentées peuvent varier d'une version à l'autre d'un SGBD. Un SGBD relationnel respectant strictement la norme SQL pourrait ainsi évoluer pour supporter plus de fonctionnalités, par exemple en ajoutant des fonctions plus récentes (fenêtres glissantes, ajouts en masse etc.), et renverrait dès lors des résultats différents selon les versions. Pour les SGBD les plus employés, le nombre d'utilisateurs garantit une rétro-compatibilité des requêtes. Pour les SGBD de moindre envergure cependant, par exemple les plus performants et récents issus de la recherche en informatique, cette rétro-compatibilité n'est pas du tout garantie.

Comme souvent en matière d'infrastructure informatique, il est donc nécessaire de tenir compte d'un compromis entre l'ancienneté et la forte adoption de certains SGBD d'une part, et les facilités et gains de performances amenées par les plus récents d'autre part. Dans le cas des données de SimFeodal, en tenant compte de cet inévitable compromis, nous avons choisi de privilégier des SGBD reconnus, soient-ils anciens et fortement adoptés ou plus récents mais utilisés par des acteurs d'envergure[35]. Ce faisant, on se coupe immanquablement de solutions extrêmement intéressantes et performantes[36]. Ce choix est toutefois en la large faveur d'une meilleure garantie de pérennité, et de robustesse en générale, des données de SimFeodal.

34. Dans les faits, on notera tout de même qu'il existe plusieurs normes successives, des « révisions » du SQL, qui apportent chacune leur lot de subtilités dans l'usage du langage. Les SGBD interrogeables en SQL ne disposent donc pas toutes des mêmes fonctionnalités, selon la révision du SQL qu'elles respectent.

35. La liste des solutions envisagées, ensuite comparées à l'aube de leurs performances, est visible dans l'axe des ordonnées de la figure 5.9.

36. Par exemple BlinkDB (AGARWAL et al. 2013), qui permet de limiter une requête à un temps maximal d'exécution donné : quand la requête n'est pas complète, le SGBD renvoi une estimation du résultat, estimation qui gagne en précision quand on augmente la limite temporelle. Un SGBD de ce type serait extrêmement précieux en *visual analytics*, mais la jeunesse de cet outil ainsi que sa nature de projet de recherche rendent incertain la continuité de son développement dans le temps. En 2018, le projet semble d'ailleurs avoir été abandonné...

Performance des SGBD Une fois que les solutions disponibles ont été discriminées par leur type, par leur interface avec les requêtes et par leur robustesse, la quantité de SGBD restant demeure de l'ordre de la dizaine. Pour choisir, parmi ceux-là, le SGBD qui sera le plus adapté aux besoins identifiés, il est donc nécessaire d'établir des critères plus précis et quantifiables. Dans le cas d'une application interactive, c'est-à-dire où le nombre de requêtes émises au cours d'une session d'utilisation peut être importante, les performances des SGBD constituent un critère majeur pour départager l'ensemble des SGBD considérés.

Il est difficile de qualifier les « performances » d'un SGBD : on entend en fait par ce terme un vaste ensemble hétérogènes de propriétés. On peut par exemple juger les performances par le filtre de la mémoire occupée par le stockage d'une base de données, ou encore par le nombre de requêtes concurrentes que peut gérer un SGBD, ou encore par la capacité à paralléliser le stockage sur plusieurs serveurs. Dans notre cas, ces points sont assez peu significatifs : en dépit de la quantité de sorties, l'ordre de grandeur – quelques gigaoctets de données – reste largement entreposable sur un environnement classique, sans besoin de parallélisation. De la même manière, SimEDB est un environnement dédié à des utilisateurs experts, en petit nombre : les chercheurs travaillant autour de SimFeodal. La quantité de requêtes simultanées ne peut donc pas dépasser la dizaine, ce qui constitue une trivialité pour l'ensemble des SGBD relationnels classiques. On s'attachera donc à juger les performances en matière de rapidité d'exécution des requêtes. Il ne s'agit pas ici de choisir un SGBD qui ferait gagner quelques millisecondes par rapport à un autre, mais plutôt de discriminer les SGBD présentant une durée de réponse trop importante pour notre usage.

En effet, plus les données sont massives, plus le temps d'exécution d'une requête augmente, souvent sous la forme d'une fonction puissance. Si tous les SGBD présentent des vitesses acceptables et proches sur des bases de données de faible volume, l'écart s'accroît considérablement à mesure que les données s'accumulent. La figure 5.9 [37] montre les différences incontestables qui existent entre les SGBD étudiés. On peut y constater que l'écart est gigantesque, par exemple vis-à-vis du temps nécessaire à une jointure, entre les 4 secondes de MonetDB et les 300 secondes (5 minutes...) de SQLite. Le choix d'un SGBD selon ses performances a donc un impact majeur sur la fluidité d'une application d'exploration de données massives. Pour départager les SGBD, on comparera leurs performances selon les différents types d'opérations demandées, qu'elles concernent l'écriture dans la base (insertion) ou des types de lecture (agrégation et jointure).

37. Dans cette figure, on compare la rapidité de différentes requêtes sur un jeu de données identique selon les SGBD. Ce type de test de performance permettant la comparaison de solutions techniques diverses est appelé *benchmark*. Ce jeu de données, composé de 100 Millions de lignes et de deux colonnes numériques, présente une volumétrie comparable (franchement inférieure en nombre de colonnes toutefois) à celle des données issues de SimFeodal qui sont interrogées dans SimEDB.

Performances en écriture et en lecture La figure 5.9 affiche des résultats qui semblent globalement ordonnés (les quatre premiers SGBD sont par exemple quasiment toujours plus lents que les 2 derniers), mais fluctuent cependant à la marge selon les opérations demandées. La première colonne du graphique montre ainsi le temps nécessaire à l'insertion du jeu de données exemple (voir la note de bas de page 37) dans le SGBD depuis un fichier CSV. Les deux colonnes suivantes exposent le temps nécessaire au traitement d'une requête, donc à une interrogation des données une fois archivées dans les bases de données. On peut constater que le classement des SGBD varie faiblement en lecture, et de manière assez faible en insertion : les deuxièmes et troisièmes colonnes respectent un ordre globalement similaire, assez différent de celui de la première colonne. Dans un environnement classique, la performance d'insertion de données est un facteur prépondérant : quand de nouvelles données sont ajoutées constamment, par exemple pour stocker des données issues de capteurs automatiques, l'insertion peut vite constituer le goulot d'étranglement de la solution.

Pour SimFeodal, l'insertion n'est pas véritablement un enjeu : les données sont ajoutées par bloc, manuellement, une fois que des nouvelles simulations ont été exécutées. C'est donc au pire un acte quotidien, mais dans ce cas, que la requête demande 10 secondes (MapD) ou 10 minutes (MySQL InnoDB), cela n'a que peu d'impact. La première colonne est donc un indicateur de performance peu adapté dans notre cas.

Les deux colonnes suivantes, relatives à l'interrogation de données, se révèlent au contraire extrêmement importantes : à chaque action de l'utilisateur de SimEDB, une nouvelle requête est envoyée pour calculer un nouvel indicateur correspondant au jeu de données filtré manuellement (cf. section 5.2.5). À chaque affichage d'onglet, une nouvelle requête est donc émise et traitée. Même si tous les indicateurs ne sont pas systématiquement mobilisés – et donc calculés – (cf. chap 3), cela signifie tout de même que pour chaque sélection, une bonne dizaine d'indicateurs seront observés, et donc, autant de requêtes. Quand une requête demande 60 secondes (par exemple PostgreSQL en « jointure »), cela implique que chaque indicateur requiert au moins une minute avant de s'afficher. Pour observer une dizaine indicateurs, l'utilisateur devra donc attendre patiemment une dizaine de minutes, sans même tenir compte du temps qu'il passera à les analyser visuellement.

Pour noircir le trait, notons de plus que les résultats communiqués dans la figure 5.9 correspondent à des requêtes simples qui ont valeur d'exemples de base. Dans le cas de SimEDB, le calcul des indicateurs requiert des requêtes plus complexes, faisant appel à des agrégations et à des jointures en même temps, et les délais affichés dans ce *benchmark* sont donc bien inférieures aux durées éprouvées en conditions réelles au sein de SimEDB.