

ou encore sur le langage Python tels que Django ou Flask), est extrêmement flexible et performante, au prix d'un développement important. Un *framework* fournit en effet des « briques » logicielles de base – les composants –, très génériques. Ces composants de bases demandent donc une forte personnalisation et un agencement complexe afin d'arriver au résultat souhaité. La communication entre ces composants doit être entièrement prévue et implémentée, et on abouti donc nécessairement sur des projets assez importants, qui demandent une réelle expertise en développement et portent le risque d'être trop complexes pour être facilement adaptés et donc rendus génériques.

À l'autre bout du gradient de développement, on peut aussi choisir de bâtir une application à partir d'un ensemble logiciel intégré, comme Tableau, qui permet d'agencer visuellement et graphiquement des composants graphiques et leurs liens. Ces outils, très usités en informatique décisionnelle, sont extrêmement simples à prendre en main, y compris pour des « utilisateurs finaux » – analystes par exemple –. En contre-partie, ils sont moins personnalisables et configurables que des solutions plus bas niveau comme les *frameworks*, et ce sont majoritairement des logiciels propriétaires, donc non modifiables.

Entre ces deux extrêmes, quelques *frameworks* intermédiaires, souvent originaires des outils de manipulation de données plus que du monde de l'informatique décisionnelle, mettent à disposition de l'utilisateur des composants de plus haut-niveau que les « briques élémentaires ». L'interaction entre les composants y est déjà pré-conçue, tout en reposant sur une construction « depuis zéro », donc personnalisables et adaptables.

Généralement, chaque *framework* est associé à un langage de programmation : le *framework* Shiny⁵⁷ s'appuie sur le langage R, Dash⁵⁸ sur le langage Python et Escher⁵⁹ sur le langage Julia. Le choix de tel ou tel *framework* dépend certes de la maturité de chaque projet – Shiny est à ce titre très en avance –, mais surtout du langage informatique que le concepteur de l'application souhaite utiliser. Dans le cas de SimEDB, le créateur de la plate-forme est adepte du langage R (voir COMMENGES et al. 2014) et pratique le *framework* Shiny depuis plusieurs années (voir CURA 2015) : le choix d'utiliser ce *framework*, au sein d'un environnement logiciel basé sur le langage R, était donc assez évident.

Manipuler les données avec R et dplyr Les langages de programmation, et en particulier les plus utilisés en analyse de données, reposent souvent sur une architecture logicielle modulaire. Le langage constitue un cœur, autour duquel des bibliothèques logicielles (des *packages* en R) viennent ajouter des fonctionnalités. Parmi ces bibliothèques logicielles, en Python comme en R, certaines sont entièrement dédiées à la manipulation de données tabulaires – on parle alors de « Data Manipulation Language » (DML) – et permettent d'effectuer des traitements avec des approches fonctionnelles, plutôt qu'avec

Est ce que
il n'y a aussi
pour un
petit tableau
avec des
cités et
3 frameworks
et des ++ ...
critère principal
meilleur =
connaissance
du développeur

57. <https://shiny.rstudio.com/> – CHANG et al. 2015

58. <https://plot.ly/products/dash/> – PLOTLY 2017

59. <http://escher-jl.org/> – GOWDA 2018, d'après BEZANSON et al. 2014

les structures impératives plus fréquemment utilisées en programmation. En R, ces *packages* constituent de véritables écosystèmes, dotés de leur propre DSL (voir p. 64) et donc d'une grammaire de manipulation de données propre.

L'un de ces *packages*, dplyr (WICKHAM et al. 2015), s'inscrit dans un écosystème dénommé tidyverse (WICKHAM 2017), et permet ainsi de chaîner des opérations de manipulation de données en une chaîne de traitement complète, plutôt que de faire appel aux habituelles boucles de parcours de matrices propres aux langages de programmation classiques. Ce faisant, avec des opérations chaînées, qui reposent sur des « verbes » permettant d'effectuer des traitements de restructurations, de modification, de filtrage ou d'enrichissement d'une donnée tabulaire⁶⁰, on obtient un ensemble d'instructions qui forment une « phrase » de manipulation de données, exprimées donc dans la « grammaire de traitement de données » fournie par dplyr.

Cette « grammaire » s'inspire notamment du SQL, bien que beaucoup plus complète, et peut en particulier être « convertie » en SQL (figure 5.16), c'est-à-dire qu'une suite d'instructions exprimées via dplyr en R (figure 5.16a) peut être traduite en SQL (figure 5.16b), et donc envoyée et exécutée sur un SGBD.

En matière de performance, l'approche de dplyr est intéressante : toutes les opérations sont effectuées par le SGBD directement, et seul le résultat final est renvoyé à R (instruction `collect()`). Le traitement de données bénéficie donc de la rapidité d'exécution du SGBD MapD, tout en profitant de la syntaxe expressive de dplyr. De plus, cela permet de minimiser les transferts de données : en exécutant les calculs dans le SGBD, il n'est besoin que d'en renvoyer le résultat à l'utilisateur. Et ce résultat est nécessairement moins lourd que les données dont il provient. On optimise ainsi l'utilisation de bande-passante internet.

```
HombreAgregatParAnnee <- tbl(conflapb, "agregats") %>% # récupération de la table agregats de la BDD
  filter(sim_name == "S_0") %>% # Filtrer la table en ne conservant que les expériences "S_0"
  group_by(sim_name, seed, annee) %>% # Agrégation sur les observations de simulation (seed et sim_name) et par annee
  summarise(NbAgregats = n()) %>% # Calcul du nombre total de lignes pour chaque agrégation
  arrange(sim_name, seed, annee) %>% # Tri de la table selon les trois variables
  collect() # Récupération du résultat de la requête en mémoire.
```

(a) Code source R avec le *package* dplyr

```
SELECT "sim_name", "seed", "annee", COUNT() AS "NbAgregats"
FROM "agregats"
WHERE ("sim_name" = 'S_0')
GROUP BY "sim_name", "seed", "annee"
ORDER BY "sim_name", "seed", "annee"
```

(b) Traduction du code source dplyr en SQL

FIGURE 5.16 – Un exemple de manipulation de données stockées dans un SGBD depuis R. On y interroge la table des agrégats de population pour calculer le nombre moyen d'agrégats par année de simulation.

60. Les fonctions de base sont donc des « verbes », au sens où elles définissent les opérations qui seront effectuées sur les données. On peut ainsi isoler des colonnes avec le « verbe » `select`, filtrer les lignes avec `filter`, modifier une colonne avec `mutate` etc. La figure 5.16a en donne un exemple commenté et concret.

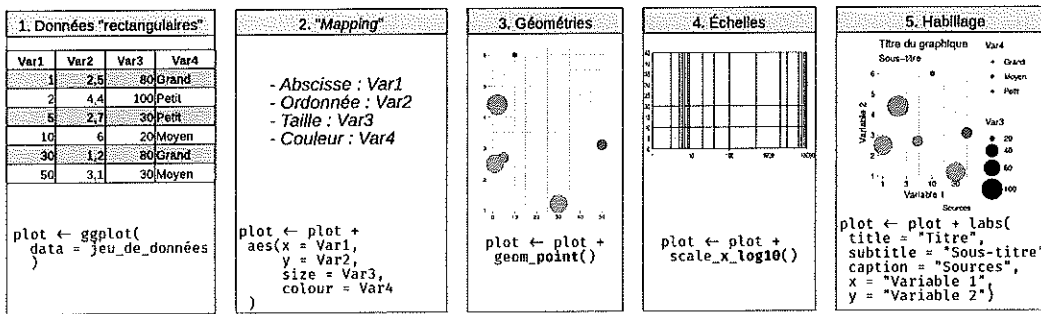
Création de graphiques avec ggplot2 et la « *grammar of graphics* »

FIGURE 5.17 – Représentation des éléments de grammaire de ggplot2, d'après une idée de HEALY (2018).

En interrogeant le SGBD avec des outils adaptés, on obtient un jeu de données qui servira de base à la représentation graphique des indicateurs (figure 5.17, étape 1). On peut alors passer à l'étape de construction graphique des indicateurs. Il existe pour cela, toujours en restant de l'environnement (et du langage) R, de nombreux *packages* dédiés.

L'un des *packages* les plus utilisés, ggplot2 (WICKHAM 2016), met en œuvre une syntaxe assez adaptée à nos contraintes : ce *package* est conceptuellement fondé sur la « *grammar of graphics* », c'est-à-dire une vision modulaire et très structurée de la conception graphique, pensée par Leland Wilkinson (WILKINSON 2006). La logique, assez familière pour un utilisateur de Systèmes d'Information Géographique (SIG), consiste à penser une représentation graphique comme un ensemble de couches (*layers*), qui se superposent, se complètent, et sont toutes basées sur une source de données. Les différentes composantes des données (variables par exemple) sont associées à des composants graphiques de base (abscisse, ordonnée, taille, couleur ...), formant ainsi une mise en correspondance (*mapping*) des données avec les composants graphiques (voir figure 5.17, étape 2).

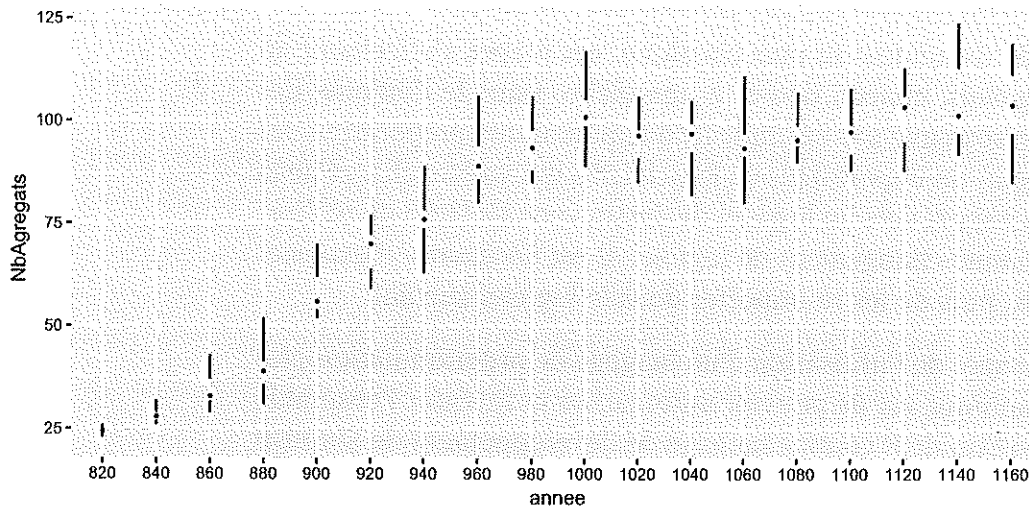
Dans notre cas, cette grammaire est porteuse d'un avantage majeur. Elle est extrêmement structurée et modulaire. Cela permet de ré-utiliser largement les codes-sources écrits pour un indicateur et de les adapter aisément à d'autres indicateurs. Si la grammaire du graphique est bien définie, elle sera ainsi très indépendante du contenu des données que l'on y insère.

Par exemple, de nombreux indicateurs de sortie de SimFeodal décrivent l'évolution du nombre d'agents au cours des années de simulation (les agrégats dans la figure 5.18). Ce type de graphique est d'une part rapide à produire avec ggplot2 : il ne requiert que quelques lignes de code (figure 5.18a). D'autre part, en changeant le tableau de données en entrée (créé dans figure 5.16a), on reproduit exactement le même type de graphique pour, par exemple, un autre type d'agent (le nombre de foyers paysans, d'églises...).

Le *package* ggplot2 répond tout à fait aux contraintes de modularité exposées plus haut, et permet de factoriser le code-source, ce qui garantit une maintenance plus rapide et une meilleure robustesse de l'application dans son ensemble.

```
ggplot(NombreAgregatParAnnee) + # création d'un graphique avec le jeu de données NombreAgregatParAnnee
aes(x = annee, y = NbAgregats) + # "mapping" des valeurs : champ "annee" en x, et champ "NbAgregats" en y
geom_violinboxplot() # ajout d'une couche graphique de type violinboxplot (à partir de la fonction ggplot2::geom_violinboxplot())
```

(a) Code source R avec le package ggplot2



(b) Graphique généré

FIGURE 5.18 – Un exemple de manipulation de données stockées dans un SGBD depuis R.

Fluidifier les étapes de rendu : le « pipeline de visualisation » DOS SANTOS et BRODLIE (2004) ont conceptualisé et schématisé l'ensemble des étapes nécessaires à la construction d'une visualisation, depuis les données brutes jusqu'à l'image finale, au sein d'un « *pipeline* » de la visualisation (figure 5.19). Ils y décrivent les différents états des données en entrée et en sortie (ligne supérieure), ainsi que les traitements que ces données subissent (ligne inférieure).

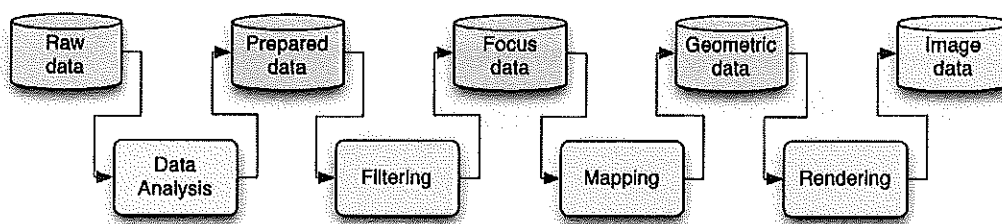


FIGURE 5.19 – « The Visualisation Pipeline », de KEIM et al. (2010), p.92, d'après DOS SANTOS et BRODLIE (2004), p. 314.

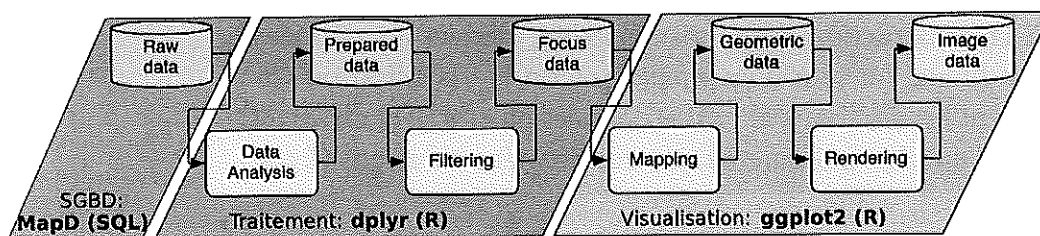
Ce *pipeline* débute par des données brutes (*raw data*) auxquelles ont fait subir un traitement (*data analysis*, par exemple une agrégation) pour obtenir des données prêtes à l'utilisation (*prepared data*). Il s'agit ensuite de filtrer ces données (choix des expériences à conserver par exemple), dont l'on conserve donc uniquement les éléments nécessaires (*focus data*). Par une étape de mise en correspondance des variables et des primitives graphiques (*mapping*, voir paragraphe précédent), on obtient un jeu de données « géométriques » (*geometric*

data). Cette « géométrie » est à entendre au sens de l'espace de la représentation graphique, qui comprend par exemple les coordonnées des points, lignes, la couleur des cercles et autres éléments mobilisés dans la construction d'un graphique. Il n'est donc aucunement question ici de données géographiques ou spatiales. La dernière étape est plus technique : il s'agit du « rendu graphique » (*rendering*), qui convertit un ensemble de spécifications géométriques (textuelles) en une image affichable, faite de pixels (*image data*).

Dans la chaîne de traitement la plus classique, ces étapes s'effectuent au sein de différents logiciels, chacun dédiés à une tâche. Dans le domaine des utilisateurs de SIG, on retrouve par exemple fréquemment une préparation des données dans un tableur, un import dans un logiciel SIG qui va être chargé de la cartographie, puis un export vers un logiciel de dessin vectoriel afin de réaliser la mise en page. À chaque changement de logiciel, il est nécessaire d'exporter les données produites, puis de les ré-importer dans le logiciel suivant.

A contrario, le propre de l'utilisation d'un langage de programmation plutôt que d'un outil graphique est de pouvoir automatiser et intégrer l'ensemble de ces étapes. L'utilisation de R comme langage de développement de SimEDB nous permet ainsi de développer une unique chaîne de traitement, qui ne requiert aucun import/export de données, et peut donc être consolidée, vérifiée et surtout ré-employée *ad libitum*.

L'enchaînement des *packages* employés dans SimEDB est présenté dans la figure 5.20a, et le code-source correspondant à l'exemple développé dans cette sous-partie dans la figure 5.20.



(a) Technologies utilisées dans SimEDB.

```
tbl(concept, "agregats") %>% summarise(n = table(hybrid)) %>% filter(
  filter(column == "5.0") %>% filter de la table en ne conservant que les hybrides 5.0
  group_hybrid(name, seed, anneel) %>% summarise(n = table(hybrid)) %>% summarise(n = table(hybrid))
  summarise(hybrid_agregats = n()) %>% filter de la table de la table pour obtenir l'agregation
  arrange(column, seed, anneel) %>% filter de la table de la table de la table de la table
  collect() %>% filter de la table de la table de la table de la table de la table
  mutate(anneel = factor(anneel)) %>% filter de la table de la table de la table de la table
  ggplot() + aes(column, y = hybrid_agregats) + geom_bar() %>% filter de la table de la table de la table
  aes(x = anneel, y = hybrid_agregats) + geom_bar() %>% filter de la table de la table de la table
  geom_turfplot() %>% filter de la table de la table de la table de la table de la table
```

(b) Implémentation d'un exemple de pipeline de visualisation pour construire un indicateur dans SimEDB

FIGURE 5.20 – Le « pipeline » de visualisation et son implémentation dans SimEDB. Cette implémentation est obtenue en assemblant les codes des figures 5.16a et 5.18a.

Modulariser les fonctions Shiny, en tant qu'outil de création d'interface graphique, bénéficie aussi d'un avantage important en matière de conception d'application web : comme ce *package* est basé sur un langage de programmation modulaire, on peut logiquement créer et ré-utiliser des « briques d'interfaces » modulaires. Par l'utilisation de modules⁶¹, il est possible de définir un ensemble d'éléments graphiques adaptatifs et de ré-utiliser tel quel cet ensemble.

Dans l'interface de SimEDB (figure 5.21), par exemple, les indicateurs graphiques sont toujours présentés de la même manière (encadrés oranges) : dans la partie de droite relative aux indicateurs, l'indicateur à proprement parler est à gauche, et des outils de téléchargement (vectoriel et image) et de notation de l'indicateur (les étoiles) sont placés en haut à droite.

En termes de code-source, la manière de produire les deux indicateurs comparés dans la figure 5.21 est strictement identique : c'est une fonction générique qui prend en entrée des données et un type de graphique à produire. Dans la figure, seul un paramètre varie : le filtre appliqué aux données, qui renvoie ici à différentes expériences. Cela permet donc d'une part de minimiser la taille du code, mais surtout, avec la généricité apportée, de faciliter de manière considérable l'ajout ou la modification d'indicateurs.

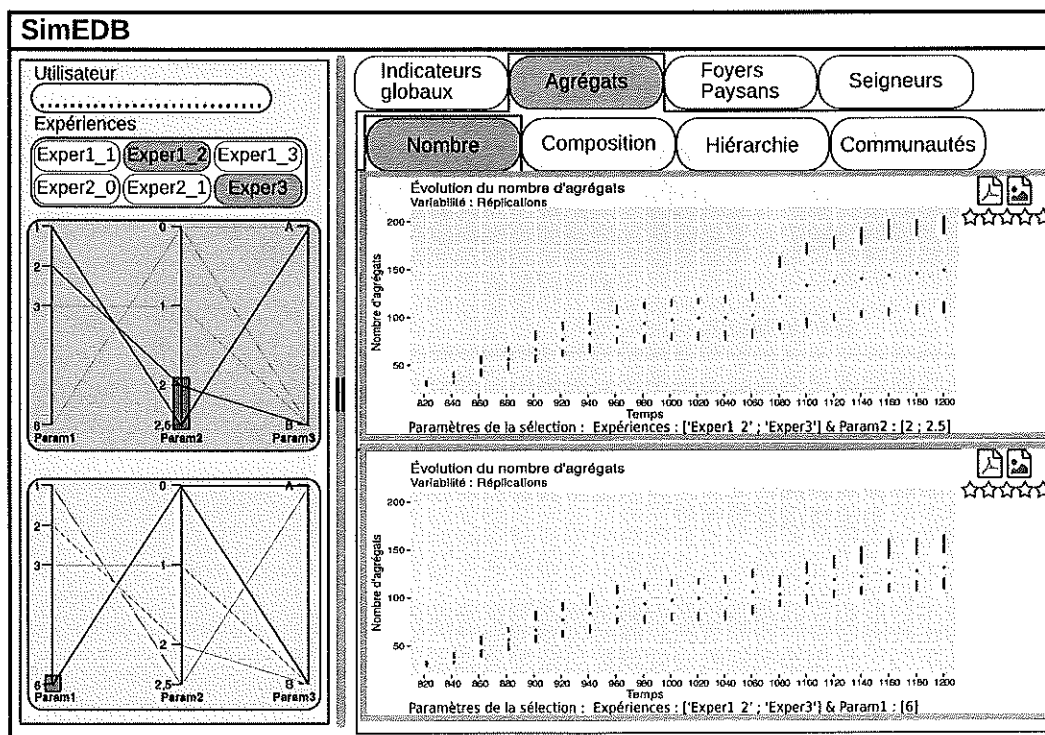


FIGURE 5.21 – Une conception modulaire. Les deux éléments graphiques encadrés sont créés par un même « module » dont les arguments varient. Ici, l'argument est constitué par les expériences sélectionnées : on le visualise dans les graphiques en coordonnées parallèles (gauche) et dans la partie inférieure des graphiques (texte « Paramètres de la sélection »).

N.B : À partir de cette figure, on représente l'interface de SimEDB à l'aide d'un *mockup* plutôt que de captures d'écrans, pour que les composants de l'interface soient plus facilement distinguables.

61. <https://shiny.rstudio.com/articles/modules.html>

5.4.2.2 Choix de l'organisation visuelle

Les différentes étapes de construction d'une plate-forme d'exploration (section 5.2 : Comment explorer les sorties de SimFeodal ?) ont conduit à une organisation sous forme de *dashboard* interactif. La forme de ce *dashboard* a évolué tout au long de l'apparition de nouveaux besoins, pour aboutir sur une organisation mono-page, pensée autour de la consultation d'indicateurs de sorties, qui devaient permettre de comparer des expériences différentes sélectionnées au moyen de graphiques en coordonnées parallèles.

Le choix d'un outil dédié à la comparaison, plus qu'à la visualisation des résultats d'un unique ensemble de simulations, entraîne nécessairement des répercussions en matière de présentation visuelle – d'interface graphique – des éléments permettant de mener cette comparaison. Depuis la première plate-forme aboutie – SimVADB (figure 5.7, reprise ici en figure 5.22a) –, l'interface graphique a fortement évolué par conséquent (figure 5.22b).

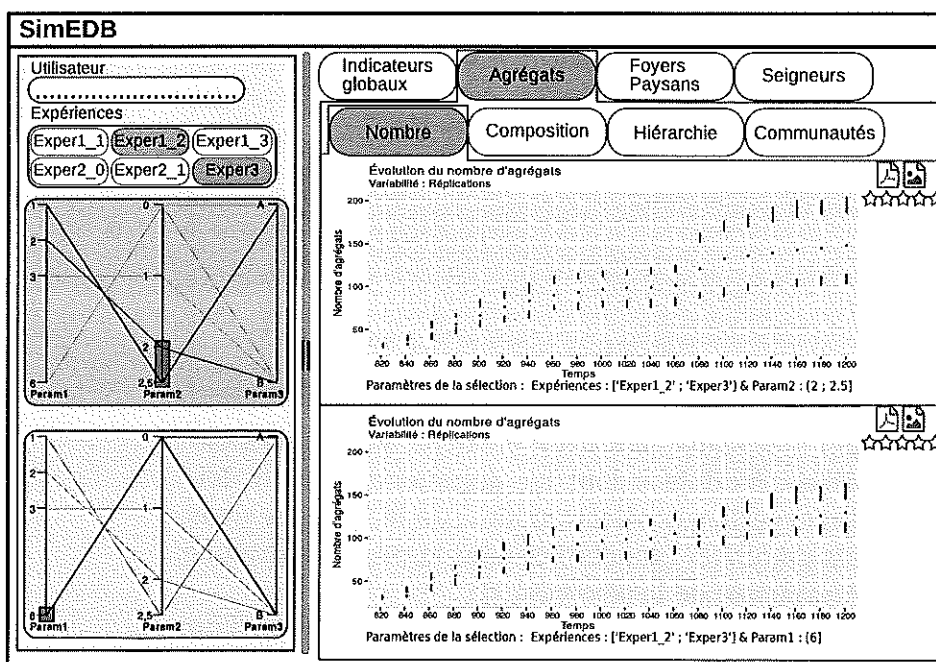
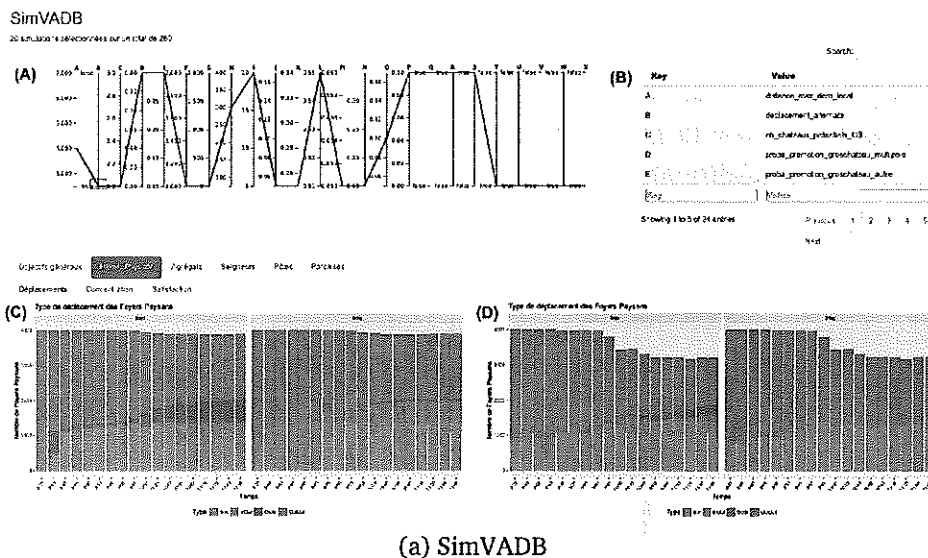


FIGURE 5.22 – Comparaison visuelle de SimVADB (a) et SimEDB (b).

Une comparaison verticale Le premier changement tient à la disposition des « contrôleurs », c'est-à-dire aux composants de l'interface sur lesquels l'utilisateur peut jouer pour choisir les expériences qui seront affichées. Dans SimVADB, le seul contrôleur était un graphique en coordonnées parallèles interactif, situé dans le haut à gauche de l'interface (figure 5.22a, partie (A)). Celui-ci permettait de régler le choix des expériences présentées dans l'indicateur de droite (D), celui de gauche (C) étant constitué d'une moyenne de l'ensemble des expériences.

Dans SimEDB, on a déjà expliqué le choix de permettre une double sélection, c'est-à-dire de régler les deux sous-ensembles de simulation à comparer. Par conséquent, la barre des contrôleurs (encadré orange dans la figure 5.23) est désormais constitué de deux graphiques en coordonnées parallèles interactifs (en bleu et en rouge), qui agissent sur les indicateurs présentés à leur droite.

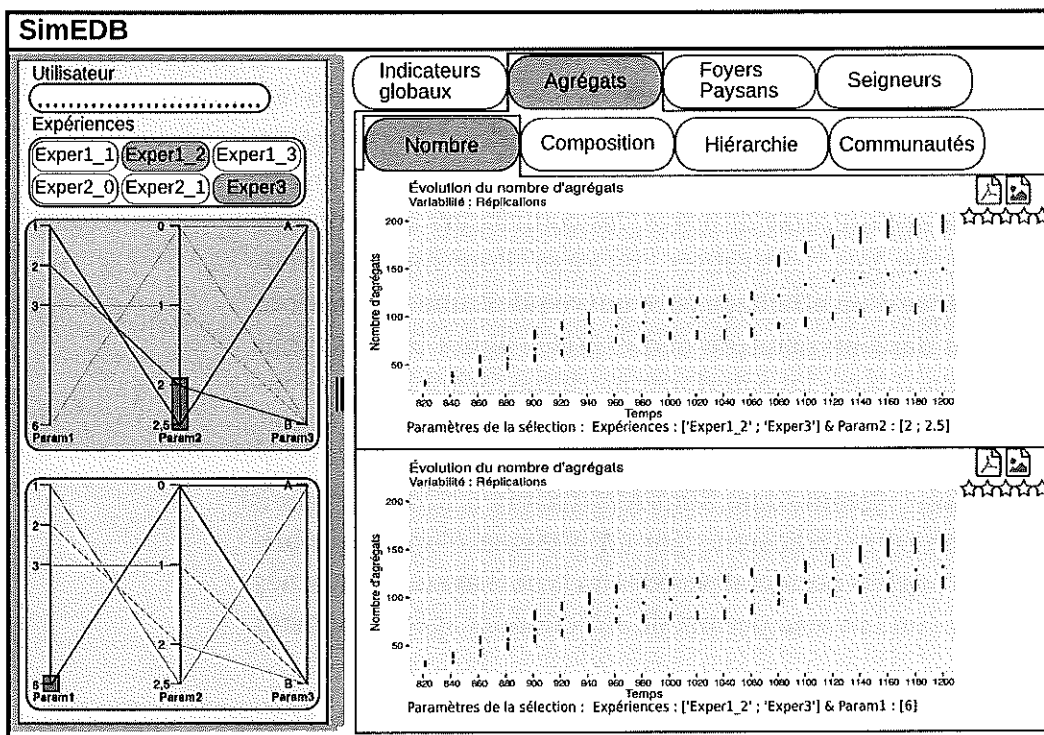


FIGURE 5.23 – La barre de contrôleurs (encadrée en orange) dans l'interface de SimEDB.

En raison de ces composants supplémentaires dans l'interface, on a choisi d'en changer la disposition. Les contrôleurs sont désormais situés à gauche et les indicateurs à droite, alors qu'ils étaient superposés dans SimVADB.

Ce choix de re-disposition tient à deux arguments. En premier lieu, en termes d'occupation de l'espace visuel on peut noter que les indicateurs graphiques sont toujours plus larges que haut. Cela s'explique notamment de manière thématique, en ce que la variation temporelle, souvent présentée en abscisse, est plus importante que les valeurs attributaires, en ordonnée : les indicateurs de ce type s'analyse plus de manière relative, en observant leur évolution, que de manière absolue. L'occupation de l'espace visuel est donc mieux employée en superposant les indicateurs : cela permet d'augmenter la taille de

chacun.

La seconde raison est méthodologique. SimEDB est un outil qui sert à comparer visuellement des données de sortie de simulation. Dans les indicateurs présentés, le plus souvent, les intervalles affichés en abscisse sont constants : qu'ils s'agissent du temps, de modalités d'un indicateur (petits seigneurs, grands seigneurs...) ou encore de seuils (composition d'un agrégat en nombre de foyers paysans, discrétisée selon des seuils choisis en amont), l'axe des abscisses est identique quelles que soient les expériences comparées. L'axe des ordonnées, au contraire, est plus variable : qu'il représente une valeur absolue, un décompte ou une fréquence, les ordres de grandeur de ces valeurs sont très hétérogènes selon les expériences comparées. Le choix a donc été fait de présenter visuellement les ordres de grandeur comparables, et donc de superposer les sélections différentes (l'axe des abscisses est alors commun) plutôt que de les afficher côte-à-côte (ce qui aurait complexifié la comparaison des ordonnées en les présentant comme directement comparables alors que les ordres de grandeur varient).

Avec un axe « fixe », il est donc opportun de mener la comparaison visuelle sur cet axe, et donc d'aligner les graphiques sur celui-ci. L'organisation des différents indicateurs est donc verticale plutôt qu'horizontale. Afin que la sélection des simulations à explorer soit intuitive, les contrôleurs doivent être alignés aux indicateurs, et dès lors, verticalisés eux aussi. Pour bien différencier visuellement ce qui relève d'un affichage et ce qui requiert une interaction, les contrôleurs s'inscrivent dans un panneau dédié, grisé (figure 5.23), ce qui constitue presque un standard dans les interfaces modernes d'applications interactives.

Onglets et sous-onglets Comme dans SimVADB (figure 5.22a), on a choisi de conserver une navigation entre indicateurs par un système d'onglets imbriqués : un premier niveau d'onglets permet d'accéder au type d'agents concernés par les indicateurs, et un second niveau permet de sélectionner spécifiquement l'indicateur choisi ⁶².

En terme de disposition, cela force l'utilisateur à interagir avec l'application régulièrement puisque chaque indicateur doit être sur un onglet dédié. La majorité des utilisateurs potentiels de SimEDB consultent toutefois l'application sur des ordinateurs portables, dotés d'écran réduits et d'une résolution faible. L'encombrement visuel est alors atteint rapidement, et mieux vaut présenter un indicateur à la fois plutôt que de présenter l'ensemble des indicateurs sur une unique page : la démarche d'étude visuelle sera plus longue, mais ne sera pas gênée ou faussée par des graphiques de dimension trop réduites qui peuvent induire des erreurs de lecture.

L'organisation des onglets en eux-mêmes pose aussi une question importante : vaut-il mieux organiser la consultation par type d'agent, ou plutôt, hié-

62. Notons que cette question revêt une importance réelle en matière d'ergonomie de l'application, mais que l'aspect technique en est pourtant assez simple. Pour changer le mode d'organisation des onglets et sous-onglets, il suffirait de réorganiser les appels aux composants dans le code-source de l'application.

rarchiquement, selon la catégorie de processus examinée (à corriger une fois fixé sur le terme, faire ref au chap3), par exemple en respectant l'ordre de consultation des indicateurs déterminé ?

Les deux approches présentent des avantages, mais nous avons choisi de rendre l'utilisation de SimEDB plus intuitive à tous, c'est-à-dire en organisant les indicateurs par type d'agents, plutôt qu'efficace, pour les utilisateurs habitués qui auraient bénéficié d'une organisation structurée hiérarchiquement.

5.4.2.3 Choix des modes d'interactions

Avant même la conception de SimEDB, avec la plate-forme SimVADB, nous avons décidé de baser la sélection des simulations sur des graphiques en coordonnées parallèles interactifs (section 5.2.5 : « Interagir avec les rapports : exploration interactive »). La logique d'ensemble du filtrage de simulations restant la même, il n'était pas nécessaire de modifier ce choix pour SimEDB.

L'accumulation d'expériences, reposant sur les variations de paramètres différents, ainsi que la démultiplication des paramètres du modèle SimFeodal ayant accompagné son paramétrage, ont pourtant demandé de reconsidérer l'usage de ces graphiques interactifs. Là où seuls quelques paramètres étaient mobilisés auparavant, les graphiques en coordonnées parallèles reposaient sur peu d'axes. Avec l'augmentation du nombre d'axes, le graphique en coordonnées parallèle est rapidement devenu illisible faute d'une surcharge graphique due au recouvrement des axes.

Réduire la surcharge visuelle des graphiques en coordonnées parallèles La première mesure pour y remédier a été de filtrer les paramètres affichés : nul besoin d'afficher un axe correspondant à un paramètre qui n'est jamais manipulé dans les expériences. Plutôt que de définir les paramètres « utiles », et donc d'avoir à les redéfinir dans l'application à chaque ajout d'expérience qui reposerait sur la variation d'un paramètre différent, nous avons fait en sorte que cette discrimination des paramètres « actifs » soit exécutée de manière automatique : quand SimEDB est lancé, une requête est exécutée sur la table des paramètres pour identifier ceux qui présentent plusieurs modalités et ceux qui n'en ont qu'une. Seuls sont alors affichés les paramètres de la première catégorie, car eux-seuls présentent un intérêt à être discriminés.

Ce faisant, le nombre de paramètres affichés est réduit, et permet d'afficher leurs intitulés plutôt que de faire appel à une table de correspondance comme dans SimVADB (figure 5.22a, partie (B)). L'automatisation de ce traitement permet de plus de ne pas avoir à changer quoi que ce soit à la plate-forme lors d'ajouts ou de suppressions de simulations de la base de données, ce qui concourt à l'objectif d'indépendance aux données de la plate-forme d'exploration.

Pré-filtrer les simulations Au fur et à mesure du paramétrage puis de la calibration de SimFeodal, les expériences ont tout de même continué à mobiliser de plus en plus de paramètres différents. Pour réduire la quantité d'in-

formation représentée et améliorer en conséquence « l'expérience utilisateur », nous avons ajouté un filtre, moins visuel que les graphiques en coordonnées parallèles, qui permet toutefois de restreindre le nombre de simulations affichées à partir de leur dénomination. Plutôt que de cibler des valeurs spécifiques de paramètres, l'idée est donc de soustraire des choix possibles des expériences entières. Pour SimEDB, on a donc ajouté un pré-filtrage, sous forme de « boîte de sélection » (*select input*, figure 5.24), qui interroge la base de données directement pour connaître les différents intitulés de simulations et agit comme un premier filtre réduisant donc les simulations interrogées dans les graphiques en coordonnées parallèles.

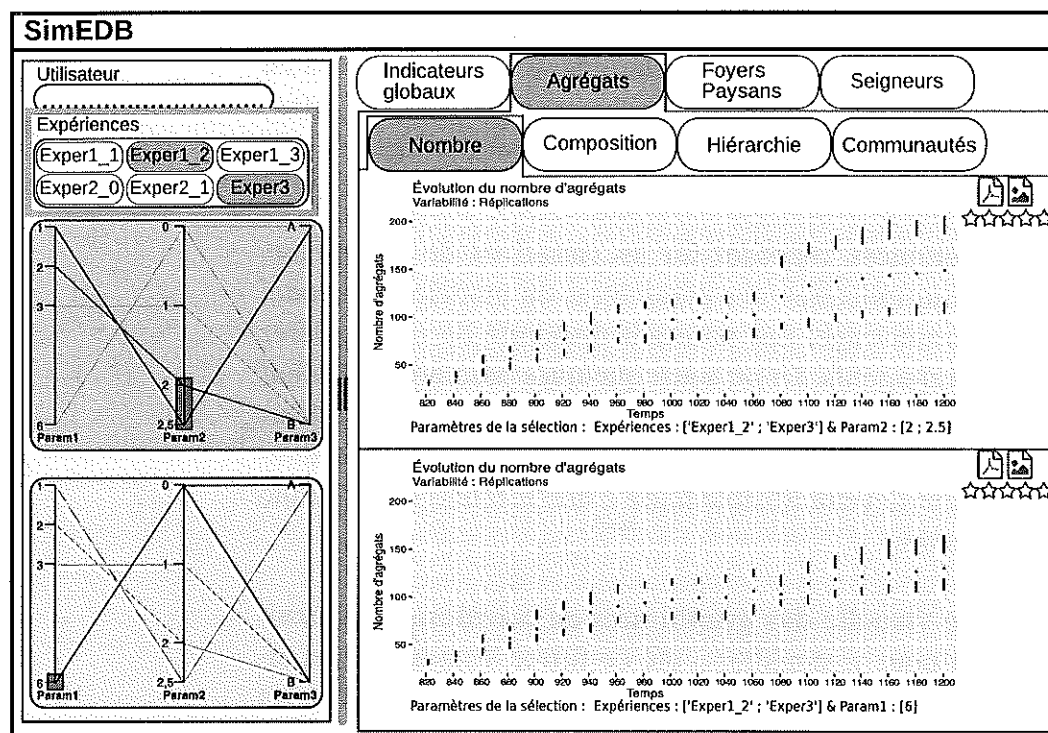


FIGURE 5.24 – Le menu de sélection des expériences (encadré en orange) qui permet un pré-filtrage des expériences à partir de leur nom.

Optimiser l'occupation de l'espace visuel En dépit de ces différentes techniques visant à minimiser le nombre d'axes affichées dans les graphiques en coordonnées parallèles, la place prise par ces graphiques reste importante, en particulier quand on décide de ne pas diminuer la taille des éléments de légende afin de conserver leur lisibilité. Quand l'application est consultée sur un écran de taille faible, l'appréhension de l'ensemble des informations présentes dans l'interface pose ainsi un véritable problème.

En réfléchissant aux séquences d'usages, par les utilisateurs, de SimEDB, on a pu comprendre que le mode d'utilisation le plus classique était de bien considérer le filtrage à effectuer sur les graphiques en coordonnées parallèles, en y consacrant un temps certain, avant de comparer longuement les différents indicateurs de la sélection. Il n'est donc que rarement fait usage de multiples filtres successifs sur un seul indicateur, dans une approche plus exploratoire donc, mais plutôt d'évaluations complètes de simulations choisies.

Il n'est alors plus indispensable de consacrer une part importante de l'es-

pace aux zones interactives (le panneau de contrôle), ou du moins, pas pendant l'ensemble de la période d'évaluation des simulations.

Un outil de redimensionnement a alors été ajouté à SimEDB, permettant, par glisser-déposer, de modifier la largeur occupée par le panneau de contrôle en l'adaptant à chaque moment au besoin de visualisation. La figure 5.25 montre ainsi une succession d'états : en début d'exploration, l'utilisateur va augmenter la taille du panneau de contrôle pour augmenter la lisibilité des graphiques en coordonnées parallèles et effectuer une sélection plus simplement. Une fois la sélection effectuée, il pourra alors re-diminuer la largeur du panneau afin d'augmenter la zone disponible pour les indicateurs de sorties de simulation, et donc la taille de ces derniers.

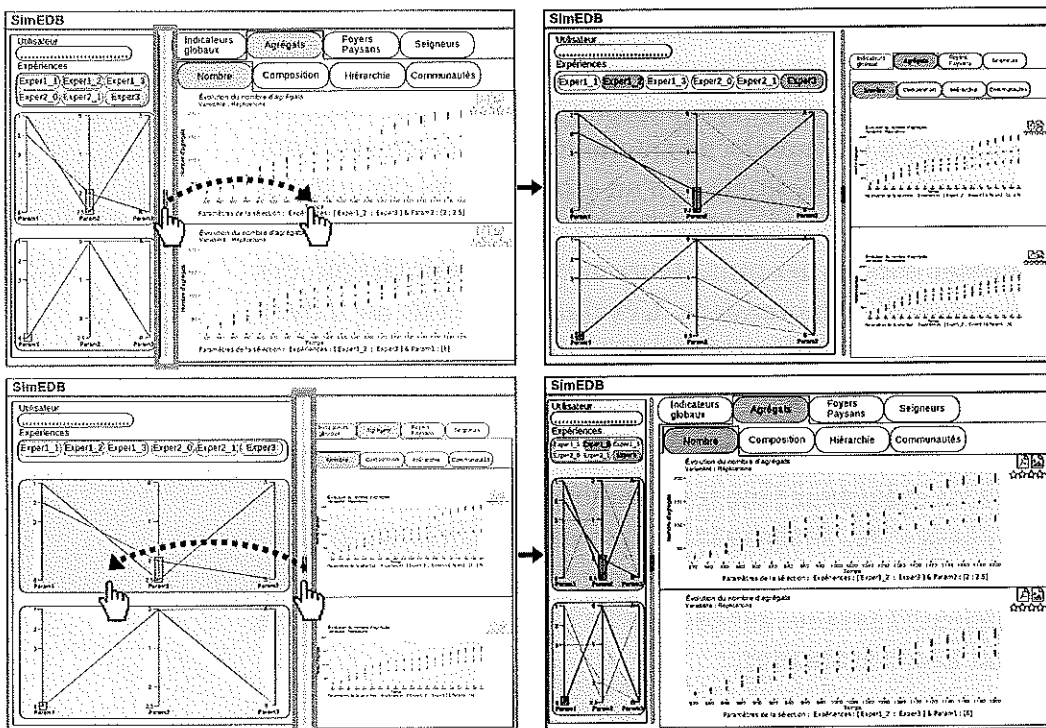


FIGURE 5.25 – Utilisation interactive de SimEDB et redimensionnement du panneau de contrôle.

Répondre aux demandes des utilisateurs : ajout d'un mécanisme d'export des indicateurs L'intérêt d'une interface modulaire et factorisée se révèle véritablement quand les utilisateurs d'un outil demandent des fonctionnalités supplémentaires, non prévues lors de la conception de l'outil. Dans le cas de SimFeodal, une telle requête est rapidement apparue : les thématiciens, mais aussi les modélisateurs, pour conserver une trace d'une session d'exploration des indicateurs, souhaitent pouvoir exporter les graphiques correspondant aux indicateurs.

Si au départ, une simple capture d'écran pouvait suffire, ce besoin a été complété par une volonté d'inclure des indicateurs de simulations dans des articles et autres communications, requérant donc des retouches des graphiques. Pour ce faire, on a choisi d'ajouter des fonctionnalités de téléchargement des graphiques, selon deux formats – image et vectoriel – afin de satisfaire à ces deux usages. Avec le développement modulaire adopté, il a suffi d'ajouter ces

fonctions d'export en un unique lieu dans le code-source de SimEDB, et l'ajout de ces nouvelles fonctionnalités a alors été disponible automatiquement pour chacun des indicateurs graphiques.

Un autre élément répond à une demande forte des utilisateurs : dans une ré-utilisation hors application des indicateurs, il était difficile de se souvenir des sélections effectuées pour produire un indicateur : en terme de reproductibilité, une fois un graphique exporté sur un ordinateur, il n'y avait plus aucun moyen de connaître les conditions précises de sa création. Nous avons donc ajouté, sur les graphiques eux-mêmes, un résumé des expériences et valeurs de paramètres sélectionnées dans chacun des indicateurs. Cela permet d'en conserver une trace plus durable et augmente leur potentiel réutilisation.

Noter les simulations Un dernier point d'interaction avec l'application a été prévu, sans pouvoir toutefois être mobilisé jusque là : il s'agissait d'aller vers une semi-automatisation de l'évaluation des simulations, par l'intermédiaire d'un outil graphique permettant de « noter » les simulations sélectionnées. Pour ce faire, et parce que, on l'a vu, l'évaluation d'un ensemble de simulations ne peut se faire de manière unique, on a choisi de donner la possibilité aux utilisateurs experts de noter chacun des indicateurs de sortie, pour chacun des ensembles de simulations qu'ils exploreraient. L'évaluation se fait au moyen d'un outil simple, composé de 5 « étoiles », et est enregistré à chaque nouvelle note.

Une piste d'utilisation serait de mobiliser les données ainsi créées, composées d'une note donnée à un indicateur pour un ensemble d'identifiants uniques de simulations, afin de réaliser des analyses quantitatives des notes attribuées : est-ce que certaines simulations sont systématiquement bien notées avec chacun des indicateurs affichés ? Certains indicateurs ne sont-ils jamais observés ou ne donnent-ils jamais lieu à évaluation ?

Cette fonctionnalité, bien qu'implémentée, n'est pas encore utilisée, mais devrait à terme permettre d'aller vers une meilleure connaissance des résultats de simulation, tout autant que vers une mesure de l'efficacité des indicateurs de sortie choisis pour évaluer un ensemble de simulations.

5.4.2.4 Présentation générale

SimEDB est une application interactive. Il nous semblait dommage d'en présenter l'utilisation d'ensemble par une succession de captures d'écran commentées. Nous avons donc réalisé une vidéo qui en montre l'usage et est disponible à cette adresse : [A faire...](#)

Conclusion

A reprendre !

Au terme de la construction de la plate-forme d'exploration, nous disposons donc d'une application, SimEDB, conçue et développée spécifiquement pour les problématiques propres à l'exploration des données de SimFeodal.

Elle s'inscrit dans les méthodes des Interactions Homme-Machine, ou même dans ce que certains nomment désormais les « Interactions Homme-Données » (« *Human-Data Interaction* », ELMQVIST 2011 ; MORTIER et al. 2014) et s'efforce de suivre les préceptes identifiés dans ce champs (AMIRPOUR AMRAII 2018, p. 167-170 par exemple).

Le développement a été fortement guidé par les contraintes et besoins identifiées, aussi bien en terme d'approches méthodologiques que de choix technologiques. SimEDB est donc un outil *ad-hoc*, toutefois pensé de manière modulaire. Tous les composants logiciels de SimEDB sont indépendants et communiquent de manière standardisée, ouvrant la voie à leur remplacement ou « interchangeabilité » : l'architecture logicielle et les choix technologiques le permettent. La plate-forme SimEDB est donc intrinsèquement pensée comme une réponse à des besoins spécifiques, mais cette réponse a été conçue comme générique et en mesure d'être adaptée aisément à d'autres types de données et/ou sorties de modèles de simulation.

Plus généralement, l'ensemble de ce chapitre montre une démarche similaire, pensée pour répondre à des besoins spécifiques avec des solutions génériques et généralisables. Le passage, depuis une succession de rapports jusqu'à une application d'exploration de ces rapports, ou encore les différents éléments relatifs au choix d'un système de gestion de base de données ou au dessin d'un modèle conceptuel de données s'inscrivent en effet dans cette même démarche qui s'ancre profondément dans une logique de recherche reproductible, aussi bien d'un point de vue technique que de celui du concept et de la méthodologie.




Références

- AGARWAL, Sameer et al. (2013). « BlinkDB : Queries with Bounded Errors and Bounded Response Times on Very Large Data ». In : *Proceedings of the 8th ACM European Conference on Computer Systems*. EuroSys '13. 00474. New York, NY, USA : ACM, p. 29-42. DOI : 10/bwrd. URL : <http://doi.acm.org/10.1145/2465351.2465355>.
- AMIRPOUR AMRAII, Saman (2018). « Human-Data Interaction in Large and High-Dimensional Data ». PhD Thesis. University of Pittsburgh.
- BATTY, Michael (2015). « A Perspective on City Dashboards ». In : *Regional Studies, Regional Science* 2.1. 00016, p. 29-32. DOI : 10/gfw9mr. URL : <https://doi.org/10.1080/21681376.2014.987540>.
- BEZANSON, Jeff et al. (2014). « Julia : A Fresh Approach to Numerical Computing ». In : 00550. arXiv : 1411.1607 [cs]. URL : <http://arxiv.org/abs/1411.1607>.
- BIMONTE, S., A. TCHOUNIKINE et M. MIQUEL (2005). « Towards a Spatial Multi-dimensional Model ». In : *Proceedings of the 8th ACM International Workshop on Data Warehousing and OLAP*. DOLAP '05. 00075. New York, NY, USA : ACM, p. 39-46. DOI : 10/cfn8w7. URL : <http://doi.acm.org/10.1145/1097002.1097009>.
- BIMONTE, Sandro (2007). *Intégration de l'information Géographique Dans Les Entreprises de Données et l'analyse En Ligne : De La Modélisation à La Visualisation*. 00029. Lyon, INSA. URL : <http://www.theses.fr/2007ISAL0105>.
- CHANG, Winston et al. (2015). « Shiny : Web Application Framework for R ». In : *R package version 0.11* 1.4. 00553, p. 106.
- COMMENGES, Hadrien et al. (2014). *R et espace : Traitement de l'information géographique*. Groupe ElementR. Lyon : Framabook.
- CURA, Robin (2015). « Créer des documents reproductibles et des applications web interactives d'analyse de données avec R : Knitr & Shiny ». http://umr5600.ish-lyon.cnrs.fr/20150610_EVS-ISIG-CafeMethodo. URL : <https://github.com/RCura/CafeMethodo>.
- (2017a). « Making Large Spatio-Temporal Data Analysis Easier. Illustrated Plea for Using (Geo)Visual Analytics. » cites : cura_making_2017. URL : <http://www.geog.leeds.ac.uk/ectqg17/home.html>.

chvisu
HACIS

- CURA, Robin (2017b). « « TimeLineEDB », application web d'exploration interactive de données de géolocalisation ». In : *M@ppemonde* 120.2015/4. URL : <https://halshs.archives-ouvertes.fr/halshs-01935702/document> (visité le 21/12/2018).
- DOS SANTOS, Selan et Ken BRODLIE (2004). « Gaining Understanding of Multivariate and Multidimensional Data through Visualization ». In : *Computers & Graphics* 28.3, p. 311-325. DOI : 10/cttwpw. URL : <http://www.sciencedirect.com/science/article/pii/S0097849304000251>.
- ELLIOTT, Roxana (2017). *How Page Load Time Affects Bounce Rate and Page Views*. Avec la coll. de SECTION.IO. cache : <https://webcache.googleusercontent.com/search?q=cache+speed-bounce-rate/+&cd=5&hl=fr&ct=clnk&gl=fr&client=ubuntu>. URL : <https://www.section.io/blog/page-load-time-bounce-rate/>.
- ELMQVIST, Niklas (2011). « Embodied Human-Data Interaction ». In : *ACM CHI 2011 Workshop "Embodied Interaction : Theory and Practice in HCI*, p. 104-107.
- EPSTEIN, Joshua M. et Robert L. AXTELL (1996). *Growing Artificial Societies : Social Science from the Bottom Up*. 05176. Washington, D.C : MIT Press. 228 p.
- FEKETE, Jean-Daniel (2010). « Infrastructure ». In : *Mastering the Information Age - Solving Problems with Visual Analytics*. Sous la dir. d'Eurographics ASSOCIATION. Eurographics Association, p. 87-108. URL : <https://hal.inria.fr/hal-00696814>.
- FEW, Stephen (2006a). *Information Dashboard Design : The Effective Visual Communication of Data*. O'Reilly Media, Inc.
- (2006b). « Multivariate Analysis Using Parallel Coordinates ». In : *Perceptual edge*. 00039, p. 1-9.
- FORCH, Valentin et al. (2017). « Are 100 Ms Fast Enough ? Characterizing Latency Perception Thresholds in Mouse-Based Interaction ». In : *Engineering Psychology and Cognitive Ergonomics : Cognition and Design*. Sous la dir. de Don HARRIS. Lecture Notes in Computer Science. 00001. Springer International Publishing, p. 45-56.
- FOTHERINGHAM, A. Stewart (1999). « Trends in Quantitative Methods III : Stressing the Visual ». In : *Progress in Human Geography* 23.4. 00038, p. 597-606. DOI : 10/c549jd. URL : <https://doi.org/10.1191/030913299667756016>.
- GOWDA, Shashi (2018). *Escher - Composable Web UIs in Julia*. JuliaGizmos. URL : <https://github.com/JuliaGizmos/Escher.jl>.
- GRIGNARD, Arnaud et Alexis DROGOUL (2017). « Agent-Based Visualization : A Real-Time Visualization Tool Applied Both to Data and Simulation Outputs ». In : *The AAAI-17 Workshop on Human-Machine Collaborative Learning*. Association for the Advancement of Artificial Intelligence 17. 00002.
- HEALY, Kieran (2018). *Data Visualization : A Practical Introduction*. 00000. S.l. : Princeton University Press. 304 p. URL : <http://socviz.co/>.
- HEINRICH, Julian et Daniel WEISKOPF (2013). « State of the Art of Parallel Coordinates ». In : *Eurographics (STARs)*, p. 95-116. DOI : 10/gd87qm. URL : <https://diglib.eg.org:443/handle/10.2312/conf.EG2013.stars.095-116>.

- IANNONE, Richard, Joseph J. ALLAIRE et Barbara BORGES (2018). *Flexdashboard : R Markdown Format for Flexible Dashboards*. 00000 R package version 0.5.1.1. URL : <https://CRAN.R-project.org/package=flexdashboard>.
- INSELBERG, Alfred et Bernard DIMSDALE (1987). « Parallel Coordinates for Visualizing Multi-Dimensional Geometry ». In : *Computer Graphics 1987*. 01538. Springer, p. 25-44.
- KEIM, Daniel et al. (2010). *Mastering the Information Age Solving Problems with Visual Analytics*. 00019. Eurographics Association.
- KITCHIN, Rob, Tracey P. LAURIAULT et Gavin MCARDLE (2015). « Knowing and Governing Cities through Urban Indicators, City Benchmarking and Real-Time Dashboards ». In : *Regional Studies, Regional Science* 2.1. 00169, p. 6-28. DOI : 10/gc92g7. URL : <https://doi.org/10.1080/21681376.2014.983149>.
- LIU, Z. et J. HEER (2014). « The Effects of Interactive Latency on Exploratory Visual Analysis ». In : *IEEE Transactions on Visualization and Computer Graphics* 20.12. 00125, p. 2122-2131. DOI : 10/f3tvrw.
- MACEACHREN, Alan M. et al. (2004). « Geovisualization for Knowledge Construction and Decision Support ». In : *IEEE computer graphics and applications* 24.1. 00256, p. 13-17. pmid : 15384662. URL : <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3181162/>.
- MACKENZIE, I. Scott et Colin WARE (1993). « Lag As a Determinant of Human Performance in Interactive Systems ». In : *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*. CHI '93. 00442. New York, NY, USA : ACM, p. 488-493. DOI : 10/dr7rj6. URL : <http://doi.acm.org/10.1145/169059.169431>.
- MORTIER, Richard et al. (2014). *Human-Data Interaction : The Human Face of the Data-Driven Society*. SSRN Scholarly Paper ID 2508051. Rochester, NY : Social Science Research Network. URL : <https://papers.ssrn.com/abstract=2508051> (visité le 17/09/2018).
- PAFKA, Szilard (2017). *Benchm-Databases : A Minimal Benchmark of Various Tools (Statistical Software, Databases Etc.) for Working with Tabular Data of Moderately Large Sizes (Interactive Data Analysis)*. 00000. URL : <https://github.com/szilard/benchm-databases> (visité le 25/08/2018).
- PANDRE, Andrew (2011). *Charts and Their Dimensionality*. URL : <https://apandre.wordpress.com/dataviews/dimensionality/>.
- PATEL, Neil (2011). *Speed Is A Killer*. 00000. URL : <https://neilpatel.com/blog/speed-is-a-killer/> (visité le 02/09/2018).
- PLOTLY (2017). *Introducing Dash*. URL : <https://medium.com/@plotlygraphs/introducing-dash-5ecf7191b503>.
- RAASVELDT, Mark et Hannes MÜHLEISEN (2018). « MonetDBLite : An Embedded Analytical Database ». In : *Proceedings of the 2018 International Conference on Management of Data*. SIGMOD '18. 00001. New York, NY, USA : ACM, p. 1837-1838. DOI : 10/gfw9mm. arXiv : 1805.08520. URL : <http://arxiv.org/abs/1805.08520>.
- REYNOLDS, Craig W. (1987). « Flocks, Herds and Schools : A Distributed Behavioral Model ». In : *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. 10133. New York, NY,

- USA : ACM, p. 25-34. DOI : 10/chhdjr. URL : <http://doi.acm.org/10.1145/37401.37406>.
- RIVARD, Kurt et Doug COGSWELL (2004). « Are You Drowning in BI Reports? Using Analytical Dashboards to Cut through the Clutter ». In : *DM Review*, <http://goo.gl/hle9Wc>. 00000.
- ROOT, Christopher et Todd MOSTAK (2016). « MapD : A GPU-Powered Big Data Analytics and Visualization Platform ». In : *ACM SIGGRAPH 2016 Talks*. SIGGRAPH '16. New York, NY, USA : ACM, 73 :1-73 :2. DOI : 10/gd7hg8. URL : <http://doi.acm.org/10.1145/2897839.2927468>.
- ROTH, Robert E. (2013). « Interactive Maps : What We Know and What We Need to Know ». In : *Journal of Spatial Information Science* 6. 00099. DOI : 10/gfw9mq. URL : <http://www.josis.org/index.php/josis/article/view/105>. 
- (2015). « Interactivity and Cartography : A Contemporary Perspective on User Interface and User Experience Design from Geospatial Professionals ». In : *Cartographica : The International Journal for Geographic Information and Geovisualization*. 00016. DOI : 10/gfw9mp. URL : <https://www.utpjournals.press/doi/abs/10.3138/cart.50.2.2427>.
- ROUMPANI, F., O. O'BRIEN et A. HUDSON-SMITH (2013). « Creating, Visualizing and Modelling the Realtime City ». In : *Proceedings of Hybrid City II 'Subtle rEvolution' Conference*. 00002.
- SCHELLING, Thomas C (1971). « Dynamic Models of Segregation ». In : *Journal of mathematical sociology* 1.2. 04241, p. 143-186.
- SHNEIDERMAN, Ben (1996). « The Eyes Have It : A Task by Data Type Taxonomy for Information Visualizations ». In : *Proceedings 1996 IEEE Symposium on Visual Languages*. IEEE, p. 336-343. DOI : 10/fwdq26. 
- SHNEIDERMAN, Ben et Catherine PLAISANT (2004). *Designing the User Interface : Strategies for Effective Human-Computer Interaction (4th Edition)*. 00000. Pearson Addison Wesley. 
- Snowflake Schema (2018). In : *Wikipedia*. 00001 Page Version ID : 856363739. URL : https://en.wikipedia.org/w/index.php?title=Snowflake_schema&oldid=856363739.
- Star Schema (2018). In : *Wikipedia*. 00001 Page Version ID : 853084099. URL : https://en.wikipedia.org/w/index.php?title=Star_schema&oldid=853084099.
- TUFTE, Edward R. (2001). *The Visual Display of Quantitative Information*. 2nd edition. 11580. Cheshire, Conn : Graphics Press USA. 190 p.
- VERMEIJ, Maarten et al. (2008). « MonetDB, a Novel Spatial Columnstore Dbms ». In : *Academic Proceedings of the 2008 Free and Open Source for Geospatial (FOSS4G) Conference, OSGeo*. 00015, p. 193-199.
- WARE, Colin (2012). *Information Visualization : Perception for Design*. 05270. Elsevier.
- WICKHAM, Hadley (2016). *Ggplot2 : Elegant Graphics for Data Analysis*. 00082. Springer.
- (2017). « Tidyverse : Easily Install and Load 'tidyverse' Packages ». In : *R package version 1.1*. 00113.

- WICKHAM, Hadley et al. (2015). « Dplyr : A Grammar of Data Manipulation ». In : *R package version 0.4 3*. 00547.
- WILKINSON, Leland (2006). *The Grammar of Graphics*. 01037. Springer Science & Business Media.
- ZAAMOUNE, Mehdi et al. (2013). « A New Relational Spatial OLAP Approach for Multi-Resolution and Spatio-Multidimensional Analysis of Incomplete Field Data ». In : *ICEIS 2013 INSTICC International Conference on Enterprise Information Systems*. 00010, p.
- ZENG, Kai, Sameer AGARWAL et Ion STOICA (2016). « iOLAP : Managing Uncertainty for Efficient Incremental OLAP ». In : *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD '16. 00012. New York, NY, USA : ACM, p. 1347-1361. DOI : 10/gfw9mn. URL : <http://doi.acm.org/10.1145/2882903.2915240>.
- ÇÖLTEKIN, Arzu, Halldór JANETZKO et Sara FABRIKANT (2018). « Geovisualization ». In : *Geographic Information Science & Technology Body of Knowledge 2018.Q2*. DOI : 10.22224/gistbok/2018.2.6. URL : <https://gistbok.ucgis.org/bok-topics/geovisualization>.