

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

www.polytech.univ-tours.fr

Projet Recherche & Développement 2015-2016

Amélioration de la plate-forme d'aide à la visualisation de données VizAssist

Tuteurs académiques
Gilles VENTURINI

Étudiants
Antoine FONFRIA (DI5)

23 mars 2016

Liste des intervenants

Nom	Mail	Qualité
Antoine FONFRIA	antoine.fonfria@etu.univ-tours.fr	Étudiant DI5
Gilles VENTURINI	gilles.venturini@univ-tours.fr	Tuteur académique, Département infomatique

Avertissement

Ce document a été rédigé par Antoine Fonfria susnommé l'auteur.

L'école polytechnique de l'université François Rabelais de Tours est représentée par Gilles Venturini susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document :

Antoine Fonfria, *Amélioration de la plate-forme d'aide à la visualisation de données VizAssist*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2015-2016.

```
@mastersthesis{
  author={Fonfria, Antoine},
  title={Amélioration de la plate-forme d'aide à la visualisation de données VizAssist},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2015-2016}
}
```

Table des matières

Introduction	1
I Recherche et Analyse	2
1 VizAssist : présentation de l'existant	3
1 Contexte, principes généraux.....	3
2 Travaux précédents.....	3
3 Technologies	4
4 Fonctionnalités	5
2 Etat de l'art	7
1 Visualisation de données.....	7
2 Sonja Rohmer : Web interfaces and assistants that support data visualization	7
2.1 Méthodologies.....	7
2.2 Récapitulatif des résultats	8
3 Mise à jour et conclusions	9
4 Survol des offres payantes.....	10
5 Autres technologies	10
3 Cahier des charges	11
1 Objectifs	11
2 Maintenance et corrections	11
3 Fonctionnalités à ajouter	12
4 Tests d'utilisabilité	13
1 Description	13
2 Préparation	14
3 Résultats des tests.....	15
4 Analyse et conclusions	17

II	Conception et Spécifications	19
5	Maquettage et conception de l'UI	20
1	F2 - Visualisation des attributs	20
2	F3 - Clipboard.....	20
2.1	Première solution - Le Clipboard.....	21
2.2	Deuxième solution - Le Dashboard.....	23
6	Spécifications	29
1	Fonctionnalités	29
1.1	F1 - Nouvelle visualisation.....	29
1.2	F2 - Visualisation des attributs.....	30
1.3	F3 - Tableau de bord	31
2	Corrections et améliorations	33
2.1	Bugs fonctionnels.....	33
2.2	Bugs d'affichage	34
2.3	Améliorations de l'UI et des visualisations.....	34
7	Organisation	36
1	Tâches et planning.....	36
2	Méthodologies et livrables	38
III	Développement	39
8	Mise en oeuvre	40
1	F1 - Visualisation Donut Chart	40
2	F2 - Visualisation des attributs	40
2.1	Histogramme d3js.....	41
2.2	Répartitions et valeurs.....	43
2.3	Info-bulles.....	45
2.4	Tri	46
2.5	Interface Utilisateur.....	47
3	Correctifs et améliorations diverses.....	47
9	Campagne de tests et débogage	52
1	Jeux de test.....	52
2	Méthode de test	52
3	Rapports de bugs	53
10	Gestion de projet et reprise	54
1	Gestion du temps.....	54
2	Procédure de versionning.....	55
3	Correctifs restants	56
	Conclusion	57

Table des figures

1 VizAssist : présentation de l'existant

1	Page d'accueil de VizAssist présentant les différentes étapes.....	5
---	--	---

4 Tests d'utilisabilité

1	Mesures relevées au cours des tests d'utilisabilité	15
2	Temps de réalisation de chaque étape par tous les testeurs	16
3	Nombre total d'erreurs faites à chaque étape	17

5 Maquettage et conception de l'UI

1	F2 - Visualisation des attributs : Maquette au crayon	21
2	Clipboard : Recherches de maquette au crayon.....	22
3	Clipboard : Boutons externes version 1, dans le coin supérieur droit de la page.....	23
4	Clipboard : Répertoire version 1A, menu déroulant intégré.....	24
5	Clipboard : Répertoire version 1B, panneau latéral superposé.....	25
6	Clipboard : Répertoire version 2, panneau latéral élargi	26
7	Dashboard : Boutons externes version 2	27
8	Dashboard : Boutons externes version 3	27
9	Dashboard : Première version	27
10	Dashboard : Version finale	28

6 Spécifications

1	F1 - Miniatures des visualisations Donut Chart nominale et ordinale	29
2	F1 - Méta-données associées au "Donut Chart"	30
3	F1 - Objectifs ciblés par "Donut Chart"	30
4	F2 - Aperçu de la visualisation graphique	31
5	F3 - Accès et transmissions au tableau de bord	32

7 Organisation

1	Planning prévisionnel des phases de développement	36
---	---	----

8 Mise en oeuvre

1	F1 - Visualisation "Donut Chart" dans l'étape 6 de l'application.....	41
2	F2 - Histogramme généré en d3js.....	41
3	Illustration du système d'échelle de d3js.....	42
4	F2 - Première visualisation des données numériques.	43
5	F2 - Affichage de l'info-bulle selon l'évènement lié au curseur.	46
6	F2 - Page des attributs avant l'implémentation de la visualisation.....	48
7	F2 - Page des attributs avec visualisation de la distribution.	48
8	F2 - Page des attributs avec visualisation des valeurs, triées selon le premier attribut.....	49
9	BA1 - Bandeau de la page d'accueil après ajout de l'étape 3.....	49
10	Am1 et Am3 - Ancienne UI de l'étape 4.	50
11	Am1 et Am3 - Nouvelle UI de l'étape 4 avec texte d'aide.	51
12	Am4 - Bouton EVALUATE en étape 6 changé en vert.....	51

10 Gestion de projet et reprise

1	Planning réel vis à vis du planning prévisionnel (tronçons noirs).....	54
---	--	----



Liste des tableaux

1 VizAssist : présentation de l'existant

1	Précédents collaborateurs au projet VizAssist.....	4
---	--	---

2 Etat de l'art

1	Plate-formes web référencées dans le document de S. Rohmer	9
---	--	---



Introduction

L'objectif de ce Projet Recherche et Développement est d'améliorer la plate-forme web d'aide à la visualisation de données VizAssist. Celle-ci a été réalisée au cours de différents projets précédents, encadrés par M. Gilles Venturini. Une version en ligne est d'ors et déjà disponible à l'utilisation. Elle permet à un utilisateur d'importer des données puis de les visualiser grâce à de très nombreux paramètres et options. Un algorithme génétique permet également de re-générer les jeux de visualisations en fonction du poids des différents attributs.

Dans la première partie de ce projet axée sur la recherche, les travaux se sont articulés en deux phases : documentation puis conception. Dans un second temps, nous verrons en détail la phase de développement.

I - Recherche et Analyse

Nous nous concentrerons ici sur la présentation détaillée de la plate-forme telle qu'elle existe au moment de la reprise : contexte, principes fondamentaux et fonctionnalités. Puis, nous ferons une comparaison de celle-ci avec les autres outils similaires disponibles, au travers de l'étude d'un document réalisé par un précédent étudiant de l'école. Ensuite, nous décrirons précisément le cahier des charges des modifications prévues pour la partie développement de ce projet : nouvelles fonctionnalités, corrections et améliorations. Enfin, le dernier chapitre présentera une série de tests d'utilisabilité suivie de leur analyse détaillée, ayant pour objectif de faire état de l'ergonomie générale de l'interface utilisateur de la plate-forme.

II - Conception et Spécifications

Dans cette deuxième partie, nous aborderons tous les travaux de conception visant à préparer le développement futur, en commençant par la conception des interfaces utilisateurs pour les fonctionnalités à rajouter et de leur intégration dans l'existant au travers de divers maquettes. Puis, un cahier de spécifications complet pour ces ajouts et corrections suivront, ainsi qu'une dernière section dédiée à l'organisation prévue pour la partie développement : planning, tâches, méthodologies, etc.

III - Développement

Cette dernière partie décrit en détail la mise en oeuvre des objectifs du cahier des charges et des spécifications. Nous verrons ainsi quelques points clés avec extraits de code à l'appui et aperçus visuels des résultats produits. Une fonctionnalité sera plutôt mise en avant avec quelques explications sur des méthodes et astuces employées. Une deuxième partie fera état de la procédure de test et de validation, puis nous terminerons par un bilan de cette phase de développement, comparant le planning prévisionnel au réel avec des informations de reprises pour de futurs projets.

Première partie

Recherche et Analyse

1

VizAssist : présentation de l'existant

1 Contexte, principes généraux

VizAssist est une plate-forme web d'assistance à la visualisation de données. Accessible depuis un simple navigateur internet, elle permet à ses utilisateurs d'obtenir une visualisation graphique de leurs données précédemment chargées. A cette fin, l'application va analyser ces jeux de données, obtenir un ensemble de méta-données ainsi qu'un certain nombre de paramétrages et autres objectifs définis par l'utilisateur, puis proposer directement les visualisations qui lui semblent les plus adaptées. L'utilisateur va alors pouvoir comparer les plus pertinentes, sélectionner la meilleure et générer différentes populations à l'aide d'un algorithme génétique.

Au delà du fonctionnel, VizAssist est développé au regard de deux grands critères fondamentaux :

- Ouverture et gratuité : La plate-forme est, et doit rester, gratuite et libre d'utilisation. Quiconque possède l'adresse du site internet peut s'en servir et cela sans aucune restrictions.
- Sécurisation des données : Aucune donnée utilisateur ne quitte l'ordinateur ce dernier et n'est propagée sur le web. Le serveur hébergeant la plate-forme n'y accède jamais et tous les calculs et rendus graphiques se font uniquement sur le poste de l'utilisateur.

Cette plate-forme qui se veut puissante et hautement configurable s'inscrit donc dans la recherche d'un outil libre et sécurisé d'aide à la visualisation des données. Elle cible ainsi toute personne ou organisme ayant ce besoin et donc un large panel d'utilisateur, de l'étudiant au salarié d'une entreprise. A cet effet, l'accessibilité et l'ergonomie sont aussi une part importante de ce qui en font les qualités.

Les outils et plate-formes de visualisation étant très nombreux, ce sont ses différents critères et qualités qui font que VizAssist possède une véritable identité, s'inscrit dans une concurrence plus restreinte et peut prétendre se positionner sur le secteur. Comme nous le verrons dans l'état de l'art ([Chapitre 2](#)), rares sont ceux qui possèdent toutes ces qualités tout en proposant des fonctionnalités similaires.

2 Travaux précédents

VizAssist a été réalisé par de nombreux collaborateurs sur plusieurs années. Initié en 2007 par Fatma Bouali et Gilles Venturini, ce dernier a encadré tous les projets étudiants depuis. Le tableau [Table 1](#) dresse la liste des précédents collaborateurs ayant participé à la réalisation du projet et quels ont été leur rôle.

Chacun d'entre eux a ainsi apporté une ou plusieurs améliorations à l'outil, que ce soit dans le fond comme l'ajout de l'algorithme génétique par Abdelheq Guettala au cours de sa thèse, ou dans la forme et les technologies comme Rémy Pradignac et Jean De Barochez qui ont transformé le projet en plate-forme web. De nombreux tests et corrections ont également été réalisés par tous au fur et à mesure de la complexification du projet.

Table 1 – Précédents collaborateurs au projet VizAssist

Date	Collaborateur	Contexte et rôle
2007	Gilles Venturini	Initialisation du projet
2007	Fatma Bouali	Initialisation du projet
2007	Noureddine Saifi	M2R : Test d'un premier modèle sans UI
2008	Abdelheq Guettala	M2R : Création d'une première interface (non web)
2009 - 2013	Abdelheq Guettala	Thèse : Ajout de l'algorithme génétique et tests
2013 / 2014	Jean De Barochez	PFE : Création de la seconde interface (web)
2013 / 2014	Rémy Pradignac	PFE : Création de la seconde interface (web)
2014 / 2015	Yann-Thomas Le Moigne	PFE : Ajouts du système de logs et autres améliorations

De plus, quelques étudiants non mentionnés dans cette liste ont apportés des ajouts mineurs comme de nouvelles visualisations à la bibliothèque de l'outil au travers de mini-projets divers.

3 Technologies

Langages

VizAssist étant une plate-forme web, elle utilise en les langages standards :

- HTML / CSS
- PHP
- JavaScript + jQuery

Les contraintes habituelles de ces langages s'appliquent donc au projet, c'est-à-dire, de manière non exhaustive : compatibilité variable selon les supports et les navigateurs, dépendance à ce même navigateur, dépendance à la connexion internet, accès restreints aux ressources matérielles, etc.

Le site fonctionnant majoritairement en local, c'est le JavaScript qui effectue les principaux calculs et génère toutes les visualisations. Les bibliothèques RequireJS et D3.js complètent alors le langage et s'ajoutent à jQuery.

Librairies JavaScript

Deux bibliothèques s'ajoutent à jQuery :

- RequireJS
- D3.js

RequireJS permet d'organiser plus efficacement les fichiers JavaScript et leur dépendances. Cela n'a donc pas d'impact direct avec les fonctionnalités du site mais facilite grandement la structuration du code.

D3.js est de son côté une bibliothèque graphique permettant l'affichage des données, c'est donc grâce à elle que toutes les visualisations du site sont générées. Très performante et disposant d'une source conséquente d'exemples et contributions amateurs, c'est un choix judicieux pour ce type de plate-forme web qu'est VizAssist.

onlinerequirejs1 onlined3js1

Outils

L'environnement et les outils de développement étant libres pour ce genre de technologies web standards et travaillant sous Ubuntu Gnome 15.10, j'ai choisi d'utiliser Atom et FileZilla pour leurs caractères open-source. Gimp est également utilisé pour tous types de manipulations d'images (création de miniatures, maquettes, ...).

Pour la gestion de projet, un tableau sous Trello permet d'épingler des tâches et d'en effectuer le suivi avec M. Venturini. Un git est installé sur le serveur et une branche a spécialement été créée pour les travaux de ce PRD, suite aux instructions de reprise contenues dans le rapport du PFE de Yann-Thomas Le Moigne.

4 Fonctionnalités

L'objectif de VizAssist étant d'aider l'utilisateur dans sa démarche de visualisation, la plupart des fonctionnalités ont été conçues dans ce but, en plus de quelques autres lui ajoutant de la valeur ergonomique ou pratique.

Etapes de Visualisation

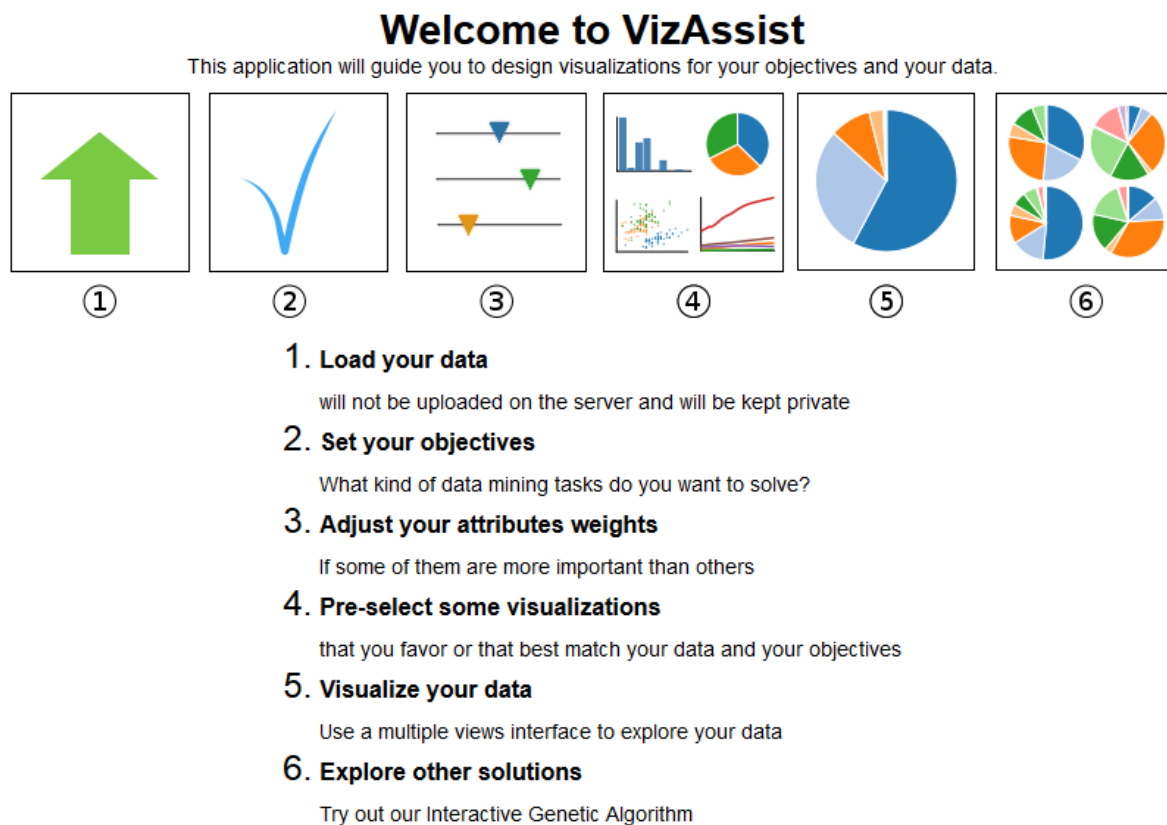


Figure 1 – Page d'accueil de VizAssist présentant les différentes étapes.

Afin de guider l'utilisateur, l'outil se décompose en étapes. Ce sera la définition admise pour ce terme dans toute la suite de ce rapport. Ces étapes, au nombre de 6, sont décrites brièvement sur la page d'accueil du site internet (**Figure 1**) avant d'être suivies rigoureusement les unes après les autres jusqu'à atteindre la visualisation finale.

Des retours sont possibles entre les étapes lorsque celles-ci ont déjà été effectuées précédemment ou atteintes (retour vers l'avant). Voici ces étapes :

1. **Chargement des données** : Chargement initial dans la version locale de l'application (téléchargement, url, copié/collé ou fichier de démo).

2. **Définition des objectifs** : Définition du type de tri à effectuer sur les données et objectifs de la visualisation.
3. **Ajustement du poids des attributs** : Pour chaque attributs du jeu de données, définition de leur importance pour les suggestions.
4. **Pré-sélection de visualisations** : Proposition des visualisations possibles avec leur pertinence et première sélection avec seulement le nom et une image d'aperçu.
5. **Visualisation** : Toutes les visualisations précédemment sélectionnées sont chargées, paramétrées en fonction des étapes 2 et 3 et sont affichables.
6. **Explorations d'autres solutions** : Algorithme génétique permettant de générer d'autres jeux de visualisations similaires au choix précédent en fonction des visualisations verrouillées au fur et à mesure.

Autres fonctionnalités

On trouve aussi d'autres fonctions venant s'ajouter aux différentes étapes. Comme par exemple la possibilité de télécharger une image de la visualisation ou la possibilité de laisser l'application détecter automatiquement des objectifs pertinents si l'on ne sait pas quoi choisir.

De plus, il existe un certain nombre de fonctions externes qui ne s'inscrivent pas dans le fonctionnement de l'outil en lui-même. On en compte 4 :

- Un Wiki contenant le manuel d'utilisation du site.
- Le Bug Report qui permet de faire remonter un problème aux administrateurs.
- Le formulaire de contact classique.
- Un formulaire d'évaluation de la plate-forme.

2

Etat de l'art

1 Visualisation de données

Si l'on souhaite faire état de la concurrence en ce qui concerne la visualisation de données, les outils et plate-formes ne manquent pas. En ce qui concerne un état de l'art, c'est plus compliqué : le domaine n'étant pas une technologie en tant que telle, de nombreux professionnels et particuliers ont produit des solutions de toute forme, dont certains ne sont ni accessibles, ni leurs démarches. On sait par ailleurs que ce qui a poussé toutes ces personnes physiques ou morales à développer ce genre d'outils, c'est un besoin lié à l'avènement des données et de leur quantité grandissante.

C'est pourquoi l'état de la concurrence fait, selon moi, un bon état de l'art en matière de visualisation de données, rassemblant ce qui fonctionne sur le marché, quelles sont les dernières innovations, les fonctionnalités les plus répandues.

Cependant, au regard des critères de VizAssist, nombre d'outils ne rentrent pas dans la même catégorie. On se restreindra donc par essence aux plate-formes gratuites du web dans un premier temps. L'élargissement aux produits et services payants peut s'avérer intéressant en terme de fonctionnalités, mais ces derniers ne peuvent pas tous faire l'objet de tests et donc de comparaisons complètes du fait de leur prix. Certains proposent des essais gratuits, mais qui sont souvent limités au niveau des actions possibles.

C'est l'étude qu'a réalisé Sonja Rohmer, doctorante ayant obtenu son Master de recherche à Polytech Tours en 2015, en comparant les plate-formes gratuites disponibles sur internet.

2 Sonja Rohmer : Web interfaces and assistants that support data visualization

Rédigé par Sonja Rohmer au cours du premier semestre 2015, *Web interfaces and assistants that support data visualization* [2] est un document faisant état des différentes plate-formes web d'aide à la visualisation de données disponibles. Il permet de les comparer à l'aide de nombreux critères, des fonctionnalités incluses et en dresse un portrait assez complet, tests à l'appui.

2.1 Méthodologies

Afin de comparer correctement les différentes interfaces de visualisation, la méthodologie mise en oeuvre a consisté à sélectionner 11 jeux de données de test correspondant aux fichiers de démonstration de VizAssist et de tenter d'importer puis d'utiliser chacun d'eux dans chaque site. En fonction de celui-ci, les formats des fichiers ont dû être modifiés pour rester compatibles.

Ces différents jeux de données ont été sélectionnés pour leur caractère hétérogène permettant de couvrir beaucoup de situation à la fois en terme de types de données, de dimensions et de quantité.

Ensuite, un certain nombre de critères de tests ont été définis et ont servi de base de comparaisons une fois l'expérience effectuée. Voici les critères retenus dans l'étude :

- Dimension des données acceptées
- Accès au site (inscription nécessaire ou non)
- Données d'entrée
 - Formats supportés
 - Données numériques
 - Données nominales
 - Données ordinales
 - Taille maximale
- Durée avant génération des visualisations
- Facilité d'utilisation
- Sécurisation des données privées
- Interactions avec les visualisations
 - Brushing¹
 - Sélection
 - Filtrage
 - Triage
 - Zoom
 - Ajout de labels

La dernière page du document propose un tableau récapitulatif des résultats obtenus pour chacun de ces critères. De plus, un résumé de l'expérience avec avantages et inconvénients a été rédigé pour chaque site.

2.2 Récapitulatif des résultats

La liste des outils que le document référence est disponible ci-dessous ([Table 1](#)). Grâce aux comparaisons effectuées, nous allons résumer en quelques points quels avantages et inconvénients ils présentent par rapport à VizAssist en fonction des différents critères d'évaluation.

Accès au site

Tous les sites nécessitent de s'enregistrer, à l'exception de VizAssist et RAW qui disposent d'un service complètement anonyme. DataWrapper et OnlineCharTool laissent aussi le loisir à l'utilisateur de choisir de s'enregistrer ou non. On retiendra que peu de plate-forme le permettent, ce qui en fait un critère d'exception pour VizAssist.

Jeux de données

Une large variété de formats d'entrées sont acceptés par les différents outils. Du format de fichier de tableur Microsoft Office Excel .xls, au format libre .csv, en passant par du simple copié-collé. Certains proposent même de charger les données depuis une url. A ce niveau, chaque site a ses propres entrées possibles mais on retiens une moyenne de deux formats acceptés, ce qui place VizAssist (.csv, url) dans le juste milieu.

On remarque ensuite que VizAssist supporte les trois types de données que sont numérique, nominal et ordinal, quand la majorité des sites comparés de prennent pas en compte le type ordinal.

Enfin, la taille des données est un critère difficile. Seul un des sites (Capsidea) arrive à afficher correctement et sans lenteurs le plus gros jeu de test. VizAssist et la plupart permettent de l'importer mais au détriment de ralentissement des performances.

Expérience utilisateur

Quatre sites sur 12 sont considérés comme rapides à utiliser, dont VizAssist, qui est également considéré comme intuitif avec 3 autres. On peut donc considérer que l'outil remplit ses objectifs de simplicité et de prise en main rapide.

1. Le brushing consiste à comparer deux visualisations d'un même jeu de données.

Table 1 – Plate-formes web référencées dans le document de S. Rohmer

Nom	URL
DataWrapper	https://datawrapper.de
OnlineCharTool	http://www.onlinecharttool.com
VizAssist	http://www.vizassist.fr
DataHero	https://app.datahero.com/get-started/welcome
RAW	http://raw.densitydesign.org
DataVisual	http://datavisu.al
Capsidea	https://capsidea.com
Silk	https://www.silk.co
Plotly	https://plot.ly/feed
manyeyes	http://www-969.ibm.com/software/analytics/manyeyes
VisualizeFree	http://visualizefree.com/moreinfo.jsp
iCharts	http://icharts.net/join

Sécurité des données

VizAssist se place dans le trio de visualiseurs avec RAW et DataVisual qui garantissent une totale sécurisation des données puisqu'elles ne sont jamais transmises sur le serveur de l'application. Tous les autres conservent les données de l'utilisateur.

Visualisations disponibles

On en compte 8 types pour VizAssist et de 7 à une vingtaine (RAW) pour les autres, avec une médiane à 11. De plus, certains outils comme DataHero en proposent certaines en 3D. VizAssist est donc plutôt à la traîne sur cet aspect.

Fonctionnalités d'interaction avec les visualisations

Cette fois, sur 6 fonctions de manipulation des données dans les visualisations (*Brushing, Sélection, Filtrage, Triage, Zoom, Ajout de labels*) VizAssist n'en possède que 3. Cela reste cependant toujours au dessus de la moyenne, où les autres plate-formes ne contiennent pas forcément les mêmes et jusqu'à deux seulement.

3 Mise à jour et conclusions

Très peu de changements ont eu lieu sur les différents outils comparés lors de l'étude, qui date de moins d'un an. Certains ont toutefois été notés, notamment pour VizAssist, la plate-forme étant sans cesse améliorée au travers des projets de Polytech.

Améliorations de VizAssist

Depuis l'étude, VizAssist supporte désormais le copié / collé de données, ce qui lui permet d'atteindre le nombre de 3 formats et le place au dessus de la moyenne pour ce critère. Par rapport au nombre de visualisations disponibles, on passe de 8 à 14 et plus si on compte certaines variantes d'un même type, ce qui propulse VizAssist au dessus de l'actuelle médiane. Toujours pas de visualisation 3D ceci étant.

Conclusions

Au regard de cette étude, on serait tenté de tirer des conclusions plutôt faciles comme un classement des meilleures plate-forme. Mais si cela n'a pas été fait, c'est justement que le nombre de critère est très important et qu'il est difficile de statuer définitivement. Il faut également garder en tête que ces critères sont là pour comparer ce qui est comparable : ce peuvent être des critères très spécifiques qui feront la force d'un outil vis à vis d'un autre, telle qu'une fonctionnalité pratique ou une ergonomie spécialement étudié.

On peut quand même retenir que VizAssist se positionne plutôt bien vis à vis des autres et n'a pas à pâlir d'un manque de choix ou d'une mauvaise expérience utilisateur. De plus, ses principes de sécurité et d'ouverture à tous ne se retrouve finalement que très peu et on peut considérer cela comme un avantage face aux concurrents.

4 Survol des offres payantes

Toute cette étude a porté sur des outils gratuits du web. La comparaison de logiciels payants et/ou à télécharger sortant du contexte, il est toutefois intéressant de regarder ce qui se fait au niveau visuel et fonctionnel. Si la concurrence ne se jouera pas entre ces deux catégories d'outils, il est tout à fait possible pour l'une de s'inspirer de l'autre, ayant le même principe : visualiser des données.

C'est notamment le cas pour le logiciel Qlik Sense : proposant un espace de travail très complet, il permet de comparer plusieurs visualisations en simultané et de les manipuler au travers d'un tableau de bord. Celui-là même qui influencera certains choix pour la fonctionnalité du même nom dans la partie Conception de ce rapport ([Chapitre 5](#)).

5 Autres technologies

En se limitant au format web, on peut déjà différencier VizAssist avec son utilisation de D3.js. Il existe en effet d'autres bibliothèques graphiques pour la visualisation et on peut notamment en retrouver une comparaison très complète sur le site de FusionCharts².

On remarque déjà que toutes ne sont pas gratuites et certaines limitent leur licence gratuite à une utilisation privée, comme HighCharts et FusionCharts. De plus, la plupart sont livrées avec des visualisations pré-fabriquées (de 6 à 13 pour les gratuites, jusqu'à 90 pour les payantes), mais seule d3.js, qui ne l'est pas, fournit une base très conséquente de plus de 200 exemples très variés. Enfin, D3.js ne produit que des images SVG quant les autres rendent leurs visualisations en HTML5, VML ou JavaScript (Google Chart Tools, Sencha ExtJS Charts, Flot).

D'autres détails vont aussi les différencier tout en restant généralement les mêmes d'un outil à l'autre, comme la qualité de la documentation (excellente pour D3.js), la permissivité du langage, les fonctions pré-intégrées, etc.

On retient donc que D3.js est un excellent choix pour un outil qui souhaite aggrandir rapidement et simplement sa bibliothèque de visualisations, grâce à de nombreux exemples et beaucoup d'aide disponibles.

2. Voir la référence webographique [[WWW1](#)]

3

Cahier des charges

1 Objectifs

Les objectifs de ce Projet Recherche et Développement sont de poursuivre la réalisation de la plate-forme, de maintenir l'existant, de le corriger et de l'améliorer. Ils s'articulent donc sur trois axes :

- Analyse de l'existant, critique et mise en évidence de dysfonctionnements, propositions d'améliorations.
- Maintenance de la version actuellement en production : reprendre des éléments de l'interface utilisateur, apporter des corrections aux problèmes fonctionnels ou ergonomiques de la plate-forme.
- Améliorer le service en ajoutant des fonctionnalités, qu'il faudra au préalable concevoir au regard de l'existant.

Le premier objectif consistant à prendre la mesure de l'existant constitue une grande part des travaux effectués au cours de la partie Recherche du projet. On retrouve à cet effet la section précédente faisant état de l'art, ainsi que toute la partie conception de ce rapport. La plupart des améliorations qui seront apportées au projet s'inscrivent aussi dans cet objectif d'apporter des propositions pertinentes, le cahier des charge définitif ayant été largement élaboré conjointement entre M. Venturini et moi même.

2 Maintenance et corrections

La maintenance et les corrections ont pour but de rendre l'application existante davantage fonctionnelle, sans en altérer ses fonctionnalités. On a donc une série de bugs et dysfonctionnements à supprimer, ainsi que des éléments de l'interface utilisateur à retravailler.

A cet effet, nous allons décomposer les tâches en regroupant les corrections en trois types distincts :

1. Bugs fonctionnels : Ce type de problème empêche une partie du programme de s'effectuer correctement. Ce peut être des événements bloquants la progression, des fonctionnalités qui ne fonctionnent pas ou des comportement anormaux du programme.
2. Bugs d'affichage : Ces problèmes correspondent à un dysfonctionnement de l'affichage des éléments de l'interface utilisateur. Ce sont par exemple des éléments qui dépassent ou sont coupés et n'apparaissent pas en entier, ou un positionnement anormal.
3. Problèmes d'UI : Ce sont des dysfonctionnements d'ordre ergonomiques de l'interface utilisateurs. Ils résultent davantage d'un problème de conception que de développement et ne sont pas des "bugs" à proprement parler. La plupart d'entre eux sont découverts en phase de tests, tels que des tests d'utilisabilité.

Toutes ces corrections à effectuer seront listés dans le cahier des spécifications. La plupart seront déterminées au cours des tests et des travaux de recherche effectués dans cette première partie du projet.

3 Fonctionnalités à ajouter

Plusieurs fonctionnalités majeures sont à rajouter au projet. En voici la liste exhaustive :

Nouvelle Visualisation

Identification : F1 - Ajout d'une nouvelle visualisation dans la bibliothèque des visualisations de la plate-forme.

Description :

L'application intègre de nombreux types de visualisations possibles pour les données. Elles y sont intégrées à l'aide d'une combinaison de 3 ajouts :

- Des méta-données pour son identification par le programme
- Des paramètres propres, incluant par exemple sa pertinence par type de donnée
- Un script D3js contenant les fonctions d'affichage de la visualisation

Pour la réalisation de cette fonctionnalité, il conviendra de trouver une nouvelle visualisation non implémenter, d'identifier ses différents paramètres, puis de l'intégrer à la bibliothèque de visualisations.

Visualisation des attributs

Identification : F2 - Visualisation de la répartition des attributs dans l'étape de définition de leurs poids.

Description :

Lors de l'étape 3 de l'application, l'utilisateur doit définir le poids de chacun de ses attributs en fonction de l'intérêt qu'il porte à leur visualisation. A l'heure actuelle, cela est possible à l'aide d'un tableau où chaque ligne ("attribut") possède un curseur permettant de faire varier le poids en pourcentage. Peu lisible et peu intuitif, il s'agit d'améliorer l'aspect visuel de cette fonction.

Pour cela, il sera intéressant de revoir la disposition de l'interface utilisateur mais surtout permettre à l'utilisateur de visualiser la répartition des données pour chaque attribut. Ceux-ci peuvent être de nature numérique ou nominal (classes), il faut donc deux types de représentation, sous forme de graphiques.

Afin de réaliser cette fonctionnalité, il conviendra de trouver une visualisation adaptée à une répartition de deux types d'attributs, d'adapter l'interface utilisateur pour l'intégrer au travers d'une phase de maquettage, puis de modifier la structure de l'étape existante au regard de la conception préalable.

Clipboard

Identification : F3 - Intégration d'un système de sauvegarde temporaire des visualisations sous la forme d'un "Clipboard".

Description :

Depuis la dernière étape de l'application, l'utilisateur doit pouvoir sauvegarder la visualisation qu'il aura paramétré au préalable. Cette sauvegarde doit être conservée le temps d'une session, et permettre le retour à celle-ci tout en travaillant sur une autre. Le terme **Clipboard**, littéralement "presse-papier", a été choisi car il suggère qu'à tout moment de la navigation, l'utilisateur doit pouvoir visualiser aisément ces sauvegardes dans un tableau de bord et éventuellement les charger pour modifications. La forme de ce dernier n'est pas défini clairement et devra donc être le sujet d'une étude approfondie lors de la conception.

Ce système de gestion de sauvegardes devra se concevoir sur deux niveaux :

- La sauvegarde, qui doit prendre en compte tous les paramètres de la visualisation en cours et s'intégrer dans la dernière étape. Le chargement d'une visualisation précédente doit également récupérer toutes ces données et permettre de retourner à l'état de l'application tel qu'il était au moment de la sauvegarde. Il faudra donc gérer la manière de conserver ces paramètres et de permettre leur manipulation par le programme.
- L'affichage du gestionnaire, permettant de visualiser les différentes sauvegardes. Son accessibilité, son apparence et son impact sur la navigation doivent être définis lors de la conception.

Le caractère temporaire de ces sauvegardes est essentiel car il s'intègre dans les notions de sécurisation des données de l'application. En effet, en conservant les données sur l'ordinateur de l'utilisateur, celui-ci en reste maître et rien n'est stocké sur le serveur.

4

Tests d'utilisabilité

Dans ce chapitre, les tests d'utilisabilités mis en oeuvre sur la plate-forme seront décrits après en avoir brièvement rappelé le concept. Nous verrons ensuite leur impact sur les spécifications en fonction de leurs résultats et analyse.

1 Description

Objectifs

L'objectif des tests d'utilisabilité est de faire un bilan provisoire ou définitif (selon l'état d'avancement d'un projet) sur le fonctionnement de son interface utilisateur. En effet, il n'est pas rare qu'une fonctionnalité utile ne soit pas, ou mal, utilisée par ses utilisateurs à cause d'une incompréhension de ces derniers. Cela passe par de plusieurs critères faisant état de son efficacité :

- Performance : Le temps ou les étapes nécessaire pour réaliser une tâche. Si celle-ci doit être réalisée régulièrement mais demande un temps considérable, on considérera que la performance est mauvaise.
- Précision : La taille d'un bouton ou sa forme peut, par exemple, provoquer des erreurs de manipulations. On parle alors ici d'une mauvaise précision de l'interface, celle-ci rendant plus complexe la réalisation d'une tâche, de manière mécanique. Ce critère prends aussi en compte la sévérité des erreurs : une fois celle-ci faite, est-il possible de revenir en arrière, et si oui, facilement ?
- Rappel : L'interface permet-elle à l'utilisateur de se souvenir suffisamment comment réaliser telle ou telle tâche qu'il a déjà faite ? Ce critère rends compte de l'ergonomie générale permettant un rappel rapide et fait par extension appel à une logique systémique des interfaces de la vie courante.
- Réponse émotionnelle : Ce dernier critère permet de rendre compte du sentiment général de l'utilisateur en fin de test. En ressort-il agacé ou neutre ? Amusé (si le projet a cette vocation) ou frustré ? Ce critère crucial impactera l'opinion de l'utilisateur et fatalement sur ses intentions de ré-utiliser l'application.

Les tests ont donc pour vocation de balayer tous ces critères de manière plus ou moins précise. Les résultats d'un utilisateur étant très subjectifs, il est nécessaire de sélectionner un panel de testeurs suffisamment large ou ciblé. Par exemple, si une application est destinée à des experts comptables, il vaut mieux privilégier peu de personnes étant susceptibles de l'utiliser, et dans le cas d'une application grand public, un grand nombre d'utilisateurs de qualifications très variées.

Procédure

Un test d'utilisabilité peut se faire de plusieurs façon mais suit généralement la procédure suivante :

- Sélection d'un panel de testeurs.
- Elaboration d'un script.
- Réalisation du script par chaque testeur avec enregistrement du temps et des erreurs commises.
- Rassemblement des résultats et comparaison
- Analyse et conclusions

La première étape consiste, comme précisé dans la section précédente, à sélectionner des utilisateurs types. Ensuite, le script sert à guider ces derniers au travers d'une succession d'étapes à réaliser dans l'application. Celui-ci doit couvrir au maximum ses fonctionnalités afin de pousser le testeur à se servir de l'interface dans sa globalité. On peut aussi choisir de se concentrer sur une partie précise uniquement. La réalisation du script doit se faire sous une observation fine du testeur. La personne en charge des tests doit alors guider ce dernier dans le script, lui demandant de réaliser chaque étape au fur et à mesure (le testeur ne doit pas avoir connaissance du script, pour garantir de bonnes conditions de test). Il peut répondre aux questions si l'utilisateur se retrouve bloqué, mais ne doit pas intervenir en dehors de ce cadre. En tant qu'observateur, il devra mesurer le temps de réalisation pour chaque instruction, afin de vérifier le critère de précision, puis quantifier les erreurs du testeur. De plus, toute remarque ou comportement non habituel ou non voulue par l'interface doit être notifié.

Toutes ces étapes de la procédure doivent rendre compte des critères d'utilisabilité décrits dans la section précédente. Ainsi, pour satisfaire le critère de rappel, par exemple, il est intéressant de faire appel à un testeur ayant déjà eu affaire à l'application par le passé, ou prévoir l'exécution répétée d'une action précédente dans le script.

2 Préparation

Au regard des critères et de la procédure décrite dans la section précédente, voici les tests d'utilisabilité réalisés sur VizAssist, telle qu'était la plate-forme en ligne au 02/12/15.

Utilisateurs testeurs

Le test a été effectué sur un panel d'utilisateurs plutôt restreint mais à priori proche de l'utilisateur final en terme de compétences techniques. Au nombre de 4, ils ont été sélectionnés parmi les étudiants de 5ème année de Polytech. Ils sont tous familiers de l'outil informatique, d'une moyenne d'âge de 23 ans et disposent d'une formation à Bac + 4 / 5. Ayant été sélectionnés selon certaines caractéristiques afin de couvrir différentes approches de l'application, voici la liste avec les points pertinents de chacun d'eux :

- A - Réalise un projet dans la visualisation de données, par conséquent familier du domaine d'application. N'a jamais utilisé l'interface.
- B - Habitué aux outils open-source aux interfaces peu optimisées et axés sur l'efficacité fonctionnelle, ne cherche jamais longtemps pour arriver à ses fins. N'a aucune affinité particulière pour le domaine d'application et n'a jamais vu l'interface.
- C - Utilisateur n'ayant pas d'affinité particulière avec ce genre d'outils ni avec le domaine d'application. N'a jamais vu l'interface.
- D - Utilisatrice ayant déjà une connaissance générale de l'application, sans l'avoir parcouru très en profondeur. La dernière utilisation remonte à plusieurs semaines et consistait à l'affichage simple d'un fichier de démonstration.

En plus de ces derniers, un utilisateur de référence a réalisé le test :

- X - Moi-même. L'intérêt est de proposer une mesure tampon, n'ayant fait aucune erreur et le temps de réaliser chaque étape étant considéré comme minimal.

Script

Le script est composé de deux parties : les instructions 1 à 4 consistent à un parcours linéaire de l'outil de visualisation, en sélectionnant un fichier de démonstration jusqu'au téléchargement. Les instructions 5 à 7 servent à tester les critères de rappel et de précision en faisant faire des aller-retours entre les étapes de l'outil de visualisation.

L'instruction numéro 3 est décomposée en 5 étapes. Cependant, ces dernières ne sont pas communiquées au testeur : elles servent de balises pour les mesures.

1. Accès au site et début de la session
2. Sélection du fichier de démonstration "Population (time series)"
3. Visualisation de ces données sous la forme d'un graphique à 2 axes au choix.
 - (a) Choix des objectifs
 - (b) Répartition des attributs
 - (c) Pré-sélection des différents types de visualisations
 - (d) Sélection du type de la visualisation
 - (e) Manipulation de l'algorithme génétique pour choisir une visualisation finale
4. Téléchargement de la visualisation
5. Retour à l'étape de répartition des attributs, modifications et nouvelle visualisation
6. Changement de fichier de démonstration
7. Visualisation de celui-ci parmi les choix les plus pertinents proposés

3 Résultats des tests

Mesures

Le tableau (Figure 1) détaille les mesures relevées au cours des tests. Les figures suivantes (Figure 2 et Figure 3) mettent en forme les données afin de comparer les temps et nombres d'erreurs pour chaque étape et pour tous les utilisateurs.

Sont considérées comme des erreurs toute action dont l'effet est en désaccord avec l'intention du testeur. Ce sont généralement de mauvais clics. Un symbole "X" signifie que l'instruction n'a pas été suivie correctement ou a été ignorée, tandis qu'un "?" marque un blocage du testeur nécessitant l'intervention du guide.

Instructions Utilisateurs	1		2		3.1		3.2		3.3		3.4	
	t	err	t	err	t	err	t	err	t	err	t	err
X	7	-	3	-	4	-	5	-	8	-	4	-
A	20	-	15	-	46	1	38	2	22	-	10	-
B	23	-	8	-	23	-	17	-	21	-	-	X
C	36	-	10	-	91	3	9	-	22	-	17	-
D	19	-	11	1	30	3	1	X	-	? X	30	-

Instructions Utilisateurs	3.5		4		5		6		7	
	t	err	t	err	t	err	t	err	t	err
X	20	-	3	-	25	-	3	-	9	-
A	76	6	20	-	80	1	5	0	35	0
B	98	5	15	-	130	3	20	-	90	2
C	67	4	11	1	132	2	6	-	10	-
D	-	X ?	3	-	80	1	3	-	20	-

Figure 1 – Mesures relevées au cours des tests d'utilisabilité

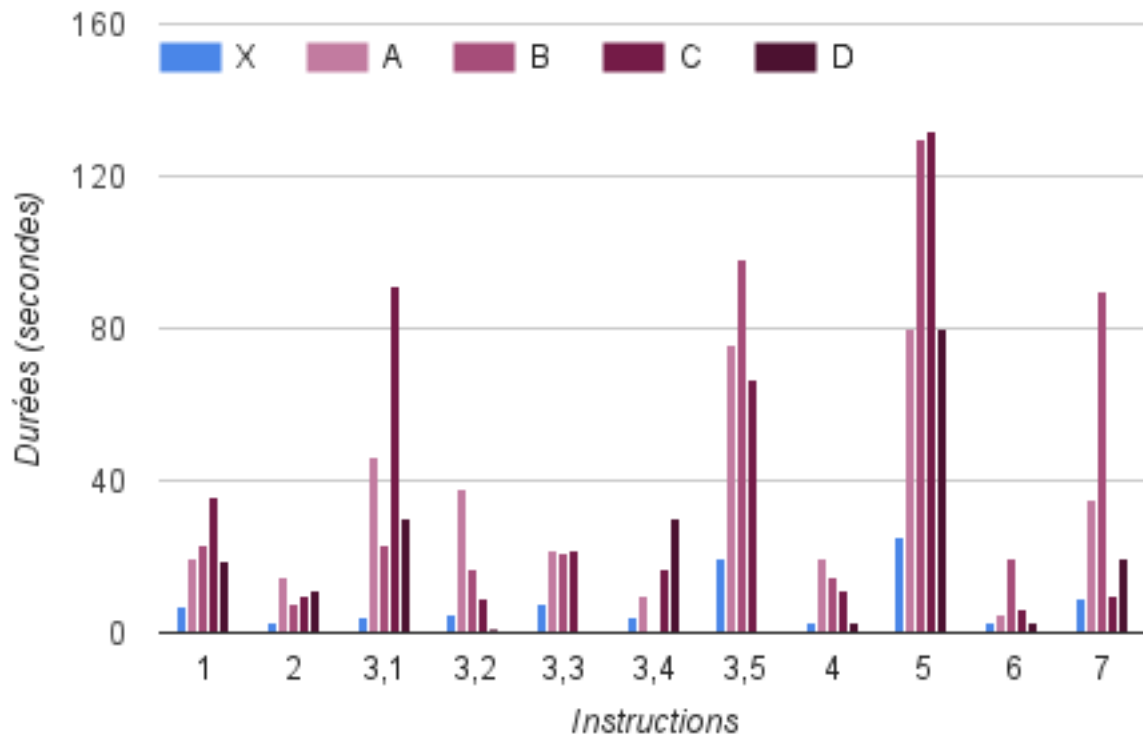


Figure 2 – Temps de réalisation de chaque étape par tous les testeurs

Observations

Chaque utilisateur a fait l'objet d'une surveillance méticuleuse pendant les tests. Sont consignés ici tout comportement anormal et erreurs significatives, ainsi que l'état émotionnel de chaque utilisateur.

Utilisateur A

- Etape 2 : Ne comprend pas ce que sont les objectifs
- Etape 3 : Confus à cause de la différence entre l'infobulle ("set your variables") dans le menu et le titre "Select your attributes".
- Etape 6 : Ne comprend pas à quoi l'algorithme génétique sert. Clique de multiples fois sur le bouton "Next Generation" sans en comprendre le fonctionnement.
- **Erreurs :**
 - Un aller-retour entre les étapes 2 et 3 par confusion
 - Clique sur "Evaluate" après l'étape 6, pensant être amené à une nouvelle étape.
- **Réponse émotionnelle :** Semble satisfait de son expérience et possède un regard critique sur les bugs qu'il a expérimenté.

Utilisateur B

- Etapes 3, 5 et 6 : Ne comprends pas le rôle du poids des attributs dans la visualisation et l'algorithme génétique.
- Etape 4 : Ne perçoit pas la possibilité de pré-sélectionner plusieurs visualisations.
- **Erreurs :**
 - Cherche à optimiser l'affichage au travers de l'algorithme génétique en sélectionnant les éléments sans comprendre que cela les verrouille.
 - Clique sur "Evaluate" après l'étape 6, pensant être amené à une nouvelle étape.
- **Réponse émotionnelle :** Réaction neutre, n'a pas d'avis particulier si ce n'est l'incompréhension face à l'algorithme génétique. Montre de la curiosité.

Utilisateur C

- Etape 4 : Ne perçoit pas la possibilité de pré-sélectionner plusieurs visualisations.
- Etape 6 : Ne comprends pas à quoi l'algorithme génétique sert.
- **Erreurs :**

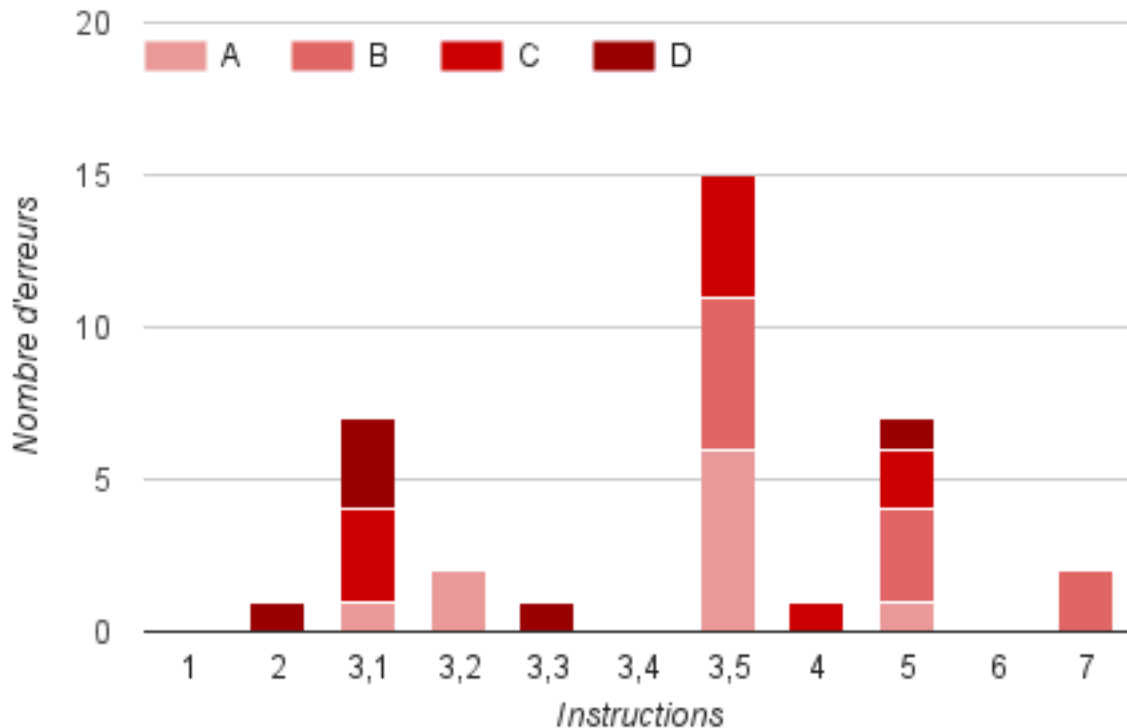


Figure 3 – Nombre total d'erreurs faites à chaque étape

- Bloque à l'étape 3.1 et clique au hasard.
- Clique sur "Evaluate" après l'étape 6, pensant être amené à une nouvelle étape.
- **Réponse émotionnelle** : Réaction neutre. Perçoit l'utilité de l'outil, ne montre pas de signes négatif en dehors de l'incompréhension de certaines fonctionnalités.

Utilisateur D

- Se perd entre les étapes, connaissant le principe général de l'outil. A du mal à se situer, se déplace trop vite.
- **Erreurs** :
 - Plusieurs aller-retours entre les différentes étapes de configuration (2, 3, 4).
 - Confond les étapes 2 et 3.
- **Réponse émotionnelle** : Montre un peu de frustration face au manque de guidage et se sent désorienté.

4 Analyse et conclusions

Analyse des données brutes

Les données recueillies mettent en avant certaines confusions et des difficultés de compréhensions. Afin d'éviter le caractère subjectif des tests, on se concentrera sur les temps importants et les erreurs résultants d'au moins deux testeurs. Dans cette optique, on remarquera tout de suite que ce sont les instructions 3.1, 3.5 et 5 qui sont à la fois les plus longues à réaliser et produisant des erreurs de la part d'au moins 3 testeurs. On note également que les valeurs du testeur D pour l'étape 3.5 n'ont pas été prises en compte car celui-ci n'a pas réussi à l'effectuer correctement.

Instruction 3.1

- Correspond à l'étape de choix des objectifs.
- Durée de référence : 4 secondes.
- Durée moyenne : 47.5 secondes, avec un écart type de 26,4 secondes.
- Nombre d'erreurs : 7 au total réparties sur 3/4 testeurs.

Instruction 3.5

- Correspond à l'étape de manipulation de l'algorithme génétique.
- Durée de référence : 20 secondes
- Durée moyenne : 80 secondes, avec un écart type de 13 secondes.
- Nombre d'erreurs : 15 au total réparties sur 3/3 testeurs.

Instruction 5

- Correspond à l'étape de manipulation de l'algorithme génétique.
- Durée de référence : 25 secondes
- Durée moyenne : 105.5 secondes, avec un écart type de 25.5 secondes.
- Nombre d'erreurs : 7 au total réparties sur 4/4 testeurs

Ces résultats mettent en lumière un dysfonctionnement de l'interface sur ces trois étapes. Les durées sont nettement supérieures à celle de référence, en particulier pour la 3.1, et le nombre d'erreur est très importante sur l'étape 3.5 quant l'un des testeur n'a pas compris comment l'achever. Ces donc sur ces trois étapes qu'il conviendra de concentrer l'étude.

Analyse des observations

Plusieurs constats s'ajoutent aux données brutes :

Tout d'abord, on relève un problème dans la localisation des étapes de l'application. Les testeurs A et D se perdent et font des aller-retours entre plusieurs d'entre elles, sans que cela en soit leur intention. La numérotation dans la barre de navigation ne suffit pas, et ce malgré l'existence d'une info-bulle qui permet aussi de relever une incohérence entre son texte et le titre des sections.

On note également que la présence du bouton "Evaluate" en fin de cheminement provoque une erreur systématique de tous les testeurs, qui se trompent, pensant accéder à une nouvelle étape.

De plus, les étapes de sélection des objectifs et des poids des attributs posent soucis pour plusieurs d'entre eux. Il semble que les instructions soient peu claires. De même que pour l'algorithme génétique, qui n'est jamais bien compris.

Enfin, à l'étape de sélection des visualisation à générer, la plupart ne perçoivent pas la possibilité d'en pré-sélectionner plusieurs. Ce n'est donc pas intuitif.

Conclusions

Grâce à cette analyse, on va pouvoir améliorer l'interface utilisateur et éviter que ces problèmes se reproduisent. Dans cette optique, voici une liste, validée par l'encadrant, des modifications à rajouter dans les spécifications :

- Les titres des sections doivent être davantage mis en valeur, de manière à mieux se situer dans les étapes. De plus, celui-ci doit être identique dans les info-bulles pour ne pas générer de confusion.
- Améliorer l'aide intégrée. Les étapes ne sont pas suffisamment claires ou explicites. Il convient donc de revoir la disposition de certains éléments ainsi que d'ajouter des explications plus claires.
 - Etapes 2 et 3 : L'explication doit être complétée et sa taille augmentée pour une meilleure mise en valeur.
 - Etape 4 : Les boutons ("SELECT ALL") doivent être déplacés en haut de la liste et une description être ajoutée.
 - Etape 6 : Les instructions ne sont pas suffisantes ni explicites. Les compléter comme pour les autres étapes.
- Le bouton "EVALUATE" en fin d'étape 6 doit être déplacé ou changé de couleur, pour ne pas confondre la sortie de l'application avec la succession d'étape que l'utilisateur vient d'expérimenter.

Deuxième partie

Conception et Spécifications

5

Maquettage et conception de l'UI

Dans ce chapitre, nous aborderons en détail la conception des éléments d'interfaces utilisateurs (UI), que ce soit pour les nouvelles fonctionnalités ou les modifications à apporter. Les deux fonctionnalités majeures à ajouter que sont la visualisation des attributs et le tableau de bord (sauvegarde) ont ainsi fait l'objet d'un travail particulier, que ce soit en terme d'ergonomie et de fonctionnel ou en terme d'intégration dans l'existant au regard de la charte graphique déjà développée. De nombreuses figures vont donc venir appuyer les explications pour ces étapes de maquettage.

1 F2 - Visualisation des attributs

L'idée derrière cette fonction étant très claire dans l'esprit de mon encadrant, il n'a pas fallu beaucoup de temps avant d'arriver sur une piste concrète. Deux éléments étaient à prendre en compte en particulier pour cet ajout : l'élargissement du tableau avec une insertion de colonne et le choix de la visualisation pour prendre en compte les deux types d'attributs, nominaux (classes) ou numériques.

Ainsi, une seule maquette réalisée au crayon (**Figure 1**). Dans cette dernière, j'ai intentionnellement inclut plusieurs possibilités de visualisations et croisé les propositions avec des petites améliorations de l'interface utilisateur. Ceci a permis de rapidement dégager le plus intéressant, en éliminant (tracés rouges) ce qui ne convenait pas.

Les propositions sur la dernière colonne n'ayant par exemple pas été retenues car jugées trop secondaires vis à vis du reste, de même que les icônes pour les types que j'avais rajouté en pensant que les termes "Nominal" et "Numeric" ne se distinguent pas assez l'un de l'autre au premier coup d'oeil.

2 F3 - Clipboard

Pour cette fonctionnalité, on peut décomposer les éléments d'interface en deux parties :

- Les boutons de gestion de sauvegarde (ajout, suppression de la visualisation dans le répertoire) qui s'intègrent dans les options déjà disponible d'une visualisation. Par la suite, ces éléments d'UI seront nommés les "**Boutons externes**".
- Le répertoire de sauvegardes permettant de voir les visualisations sauvegardées et de les charger. Pour la suite, cet affichage particulier sera nommé le "**Répertoire**".

Le maquettage de ces éléments s'est déroulé en plusieurs phases évolutives où chaque prototype a été proposé et soumis au regard de mon encadrant M. Venturini. Ensemble, nous avons alors re-défini au fur et à mesure ce que cette fonctionnalité devait être et ce à quoi elle devait ressembler à la fin. Nous avons alors dégagé ce qui faisait la force de chaque prototype pour produire la version suivante. Les paragraphes et figures suivants présentent de manière détaillée les différentes étapes de réflexions qui nous ont amenés aux spécifications finales.

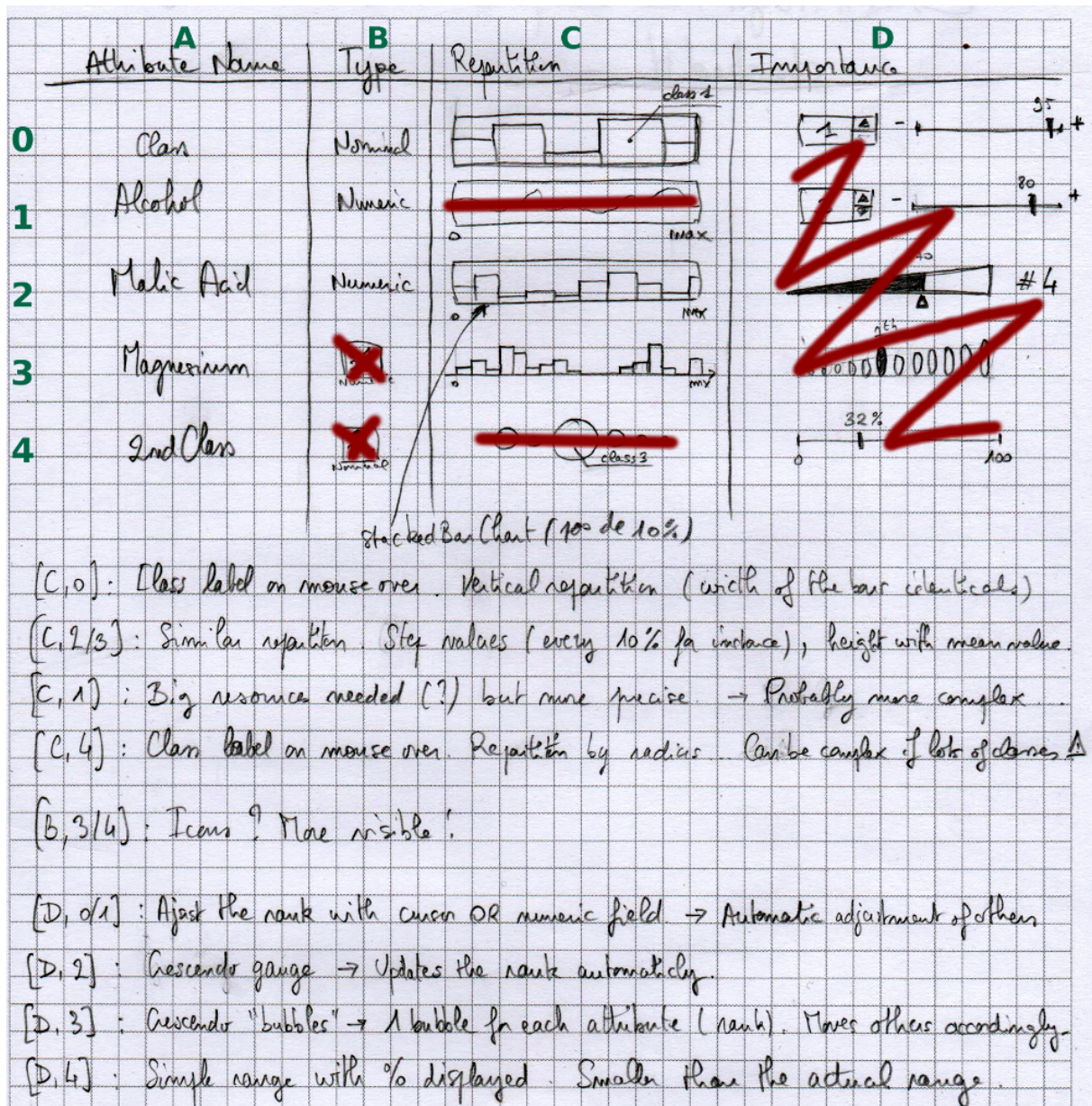


Figure 1 – F2 - Visualisation des attributs : Maquette au crayon

2.1 Première solution - Le Clipboard

La première piste qui a été explorée pour cette fonctionnalité consistait à sa plus simple définition : un gestionnaire de sauvegardes depuis lequel enregistrer, parcourir et charger les visualisations. C'est ce que nous nommons à ce moment le **"Clipboard"** : un simple répertoire de sauvegardes.

Sans plus d'éléments, j'ai débuté les recherches de propositions sur papier (Figure 2) avant d'entamer la réalisation de maquettes pour deux versions qui me semblaient pertinentes.

Boutons externes : premier placement

L'intérêt du répertoire de sauvegarde est d'y avoir accès depuis toutes les étapes de la plate-forme. Il fallait donc trouver un emplacement adéquat qui ne générerait pas le reste de l'interface utilisateur. Au vu de la disposition actuelle, le côté droit de l'affichage s'est naturellement imposé, d'autant qu'un certain nombre d'éléments s'y trouvent déjà à l'étape 5.

— Le bouton "DOWNLOAD", qui apparaît en étapes 5 et 6 et permet de télécharger la visualisation.

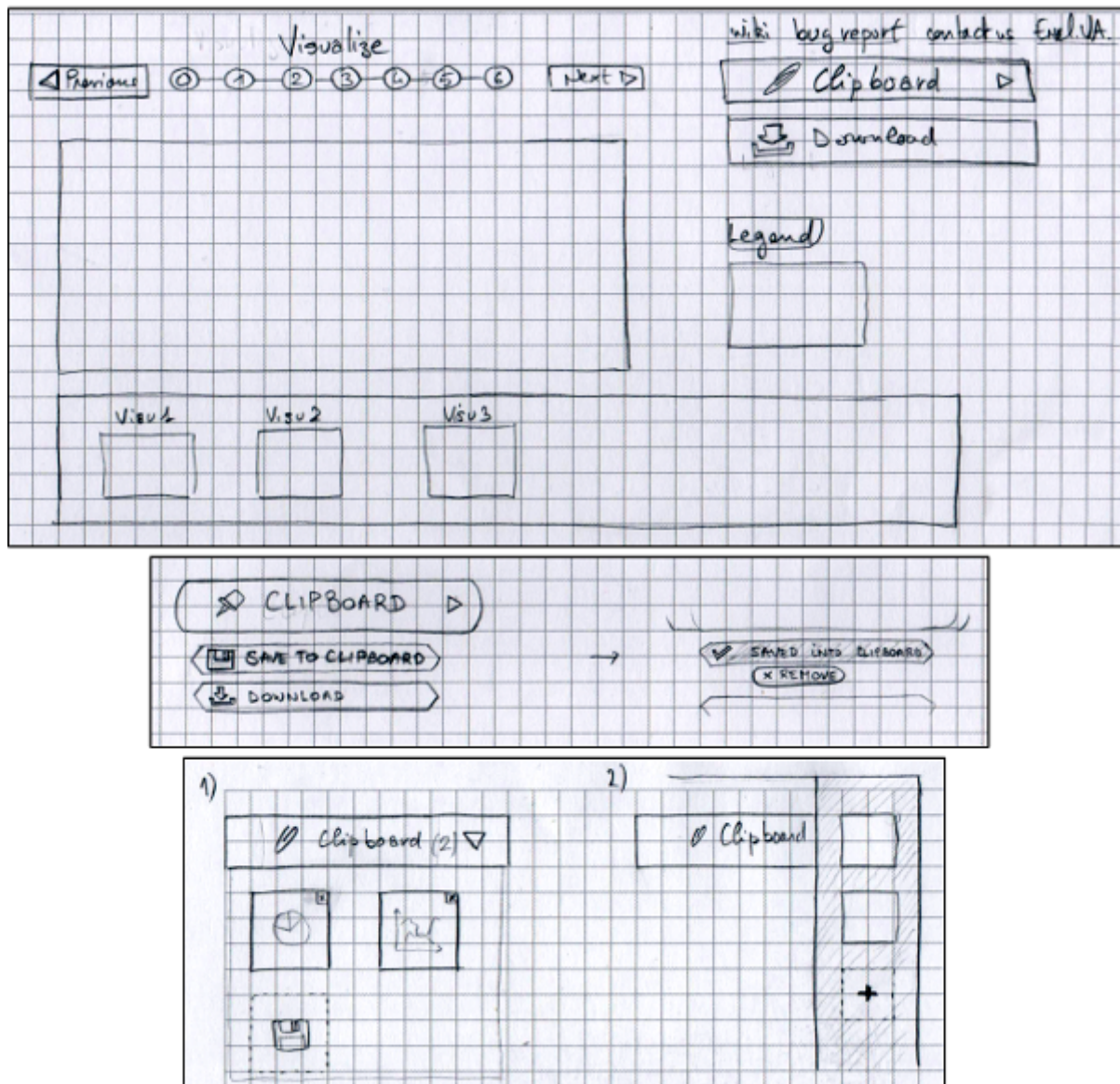


Figure 2 – Clipboard : Recherches de maquette au crayon

- La légende de certaines visualisations.

Ces éléments seront donc manipulés pour adapter le tout avec les nouveaux boutons. Pour ces premières versions, j'ai donc légèrement déplacé ces derniers pour rajouter le bouton d'ouverture comme en témoigne la figure 3.

Répertoire : deux premières propositions

Deux versions sont étudiées pour ce premier travail d'ébauches.

- Version 1A (Figure 4) - Un menu déroulant s'ouvre et est totalement intégré dans l'interface existante. Celui-ci va déplacer les éléments du dessous vers le bas et possède une taille très restreinte.
- Version 1B (Figure 5) - Un panneau latéral s'ouvre sur la droite, se superposant à l'interface existante. La taille est limitée en largeur mais sa hauteur prends la page.

La version A (Figure 4) est intéressante car permet de rester en permanence dans l'interface courante et conserve donc une continuité dans la navigation. L'autre version (Figure 5) est également intéressante car permet de s'affranchir de toute perturbation du reste de l'interface tout en ayant une hauteur maximale. Dans les deux cas, un survol des miniatures à la souris permet d'afficher le nom de la visualisation et de la charger au clic.

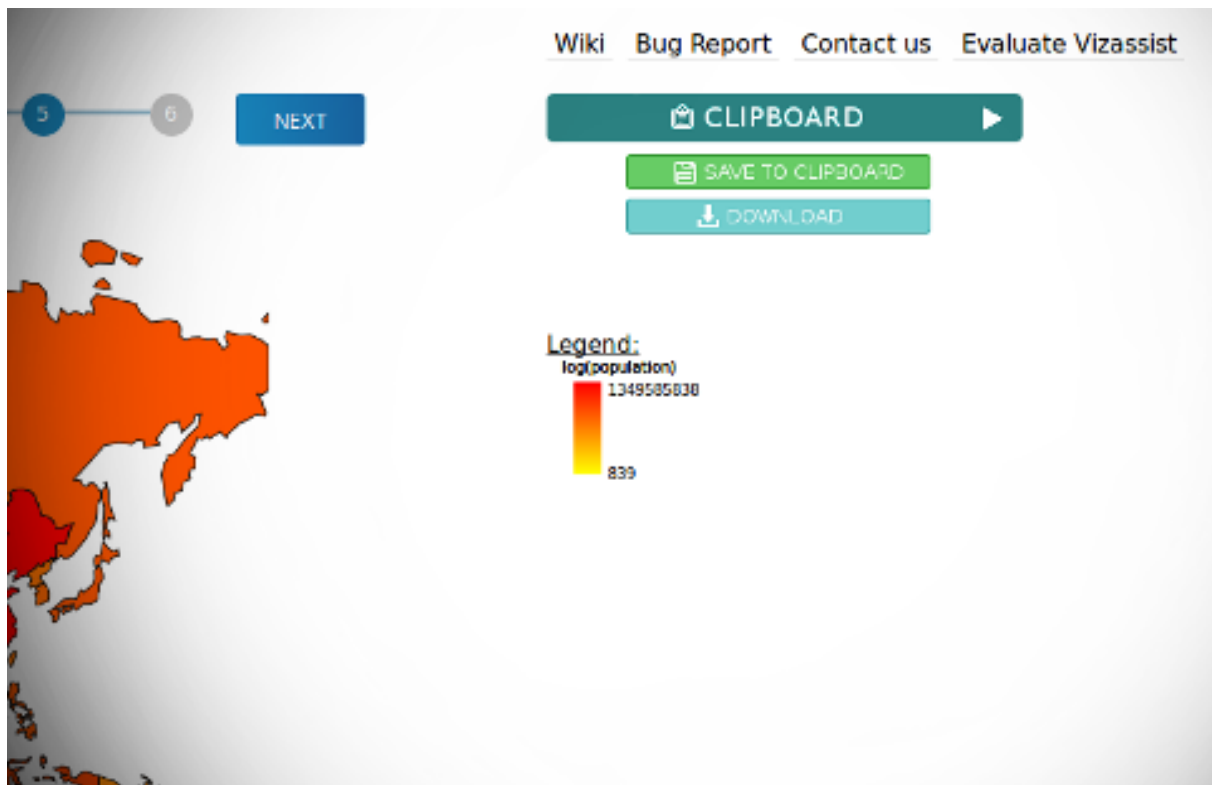


Figure 3 – Clipboard : Boutons externes version 1, dans le coin supérieur droit de la page

Après en avoir discuté, la seconde solution semblait plus adéquat : le menu déroulant ne pouvant s'étendre à l'infini, les restrictions de taille étant trop conséquentes. Je suis donc parti sur un panneau latéral superposé à l'existant. Une autre remarque était la taille des miniatures, jugées trop petites.

Répertoire : Amélioration du panneau latéral

Conformément au choix d'un panneau latéral, une seconde version est définie : on souhaite désormais avoir une vue plus élargie et un aperçu plus net de chaque visualisation (Figure 6). On a également choisi de laisser le menu principal supérieur visible en réduisant légèrement la hauteur du panneau.

Boutons externes : Seconde disposition

Le principal problème de la précédente version est l'éparpillement des éléments de l'interface. Pour résoudre cela, j'ai choisi de regrouper les boutons au niveau du menu de navigation (Figure 7). On a alors le bouton d'accès au Clipboard toujours visible aux côtés du bouton "NEXT" et garantissant ainsi une cohérence dans la structure du site. Les boutons de téléchargement et de sauvegarde étant alors placés en dessous puisqu'ils n'apparaissent qu'en étapes 5 et 6.

De plus, un indicateur (un nombre entre parenthèses) a été rajouté dans le bouton d'ouverture avec le nombre de visualisations sauvegardées. Sur la figure, ce dernier indique "DASHBOARD" et non plus "CLIPBOARD". La raison à cela est décrite dans la sous-section suivante.

Sur la figure, on peut voir les deux aspects possibles des boutons : lorsque le répertoire est fermé ou ouvert. Dans ce dernier cas, les boutons de navigations se grisent et le bouton d'ouverture change, servant alors à fermer le répertoire.

2.2 Deuxième solution - Le Dashboard

Première version

A ce moment de la conception s'est posé une nouvelle problématique. Le clipboard dans son état actuel n'allait pas suffire au regard de nouvelles exigences. En comparant les maquettes et en discutant de

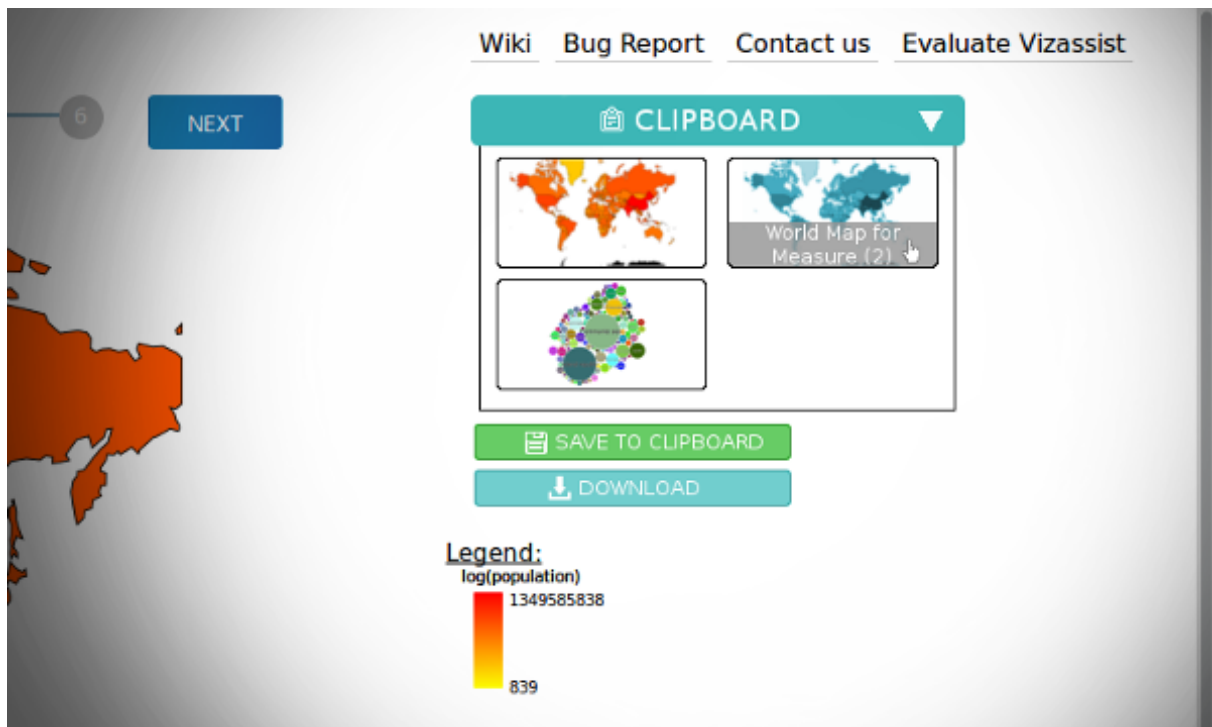


Figure 4 – Clipboard : Répertoire version 1A, menu déroulant intégré

l'utilité de la fonctionnalité, M. Venturini a eu l'idée d'améliorer le concept et de prévoir l'ajout d'un tableau de bord plus complet pour le simple clipboard. C'est-à-dire de pouvoir réellement comparer les différentes visualisations en grand format, et ne pas se restreindre à de petites miniatures sur un bord de l'écran.

A partir de ce moment, on nommera la fonctionnalité F3 comme étant le Tableau de bord (ou Dashboard) et non plus le Clipboard.

Le nouvel objectif étant donc de créer un répertoire de taille bien supérieure avec un maximum de sauvegardes affichées à l'écran et dans une instance séparée de la session en cours sur le site. J'ai donc repris les maquettes et proposé de créer non plus un panneau latéral mais un cadre venant se superposer à l'écran dans sa globalité.

La première maquette (Figure 9) de ce tableau de bord, que l'on nommera désormais le "Dashboard" propose donc un nouveau répertoire suivant ce modèle. Prenant en considération mes possibles évolutions de la fonctionnalité évoquées avec M. Venturini, j'ai pris la liberté de prévoir l'ajout d'onglets permettant de paralléliser plusieurs fenêtre au tableau de bord et d'intégrer un gestionnaire plus complet sous forme de hiérarchie de dossiers, sur la gauche, avec possibilité d'afficher de nombreuses informations.

Version finale

Finalement, l'idée du petit gestionnaire de dossiers et d'informations n'est pas retenue. De même, l'aspect général semblait trop éloigné de la charte graphique de la plate-forme. C'est enfin une version plus épurée et au visuel plus fidèle du tableau de bord que j'ai proposé à M. Venturini, qui la validera pour les spécifications (Figure 10). On retrouve alors un affichage beaucoup plus sobre où 6 visualisations peuvent être vues simultanément, sur tout l'espace disponible du navigateur. J'ai conservé l'idée de départ d'un simple survol de la souris pour afficher le bouton de chargement de la sauvegarde (visible sur la visualisation de droite de la maquette).

Cette version finale s'accompagne d'une légère révision des boutons externes (Figure 8) au niveau de la couleur de ceux-ci et de l'abandon du changement du bouton d'ouverture en "BACK" pour être simplement grisé quand le tableau de bord est ouvert. Une petite croix en haut à droite permettant de le fermer.



Figure 5 – Clipboard : Répertoire version 1B, panneau latéral superposé



Figure 6 – Clipboard : Répertoire version 2, panneau latéral élargi

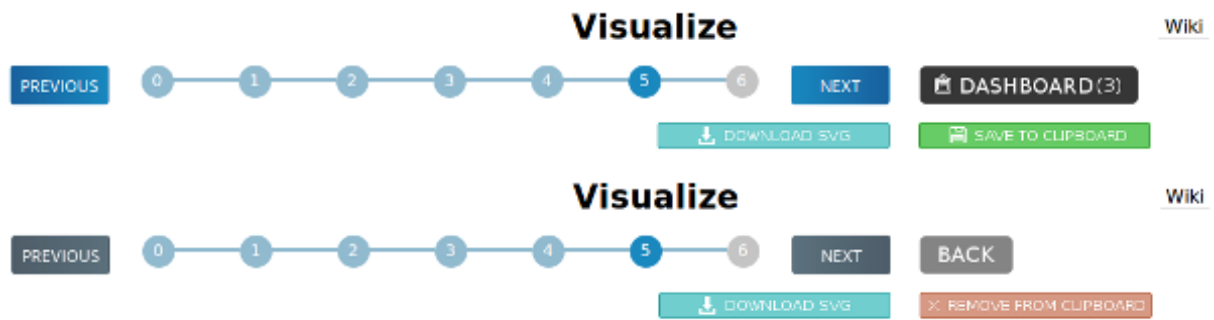


Figure 7 – Dashboard : Boutons externes version 2

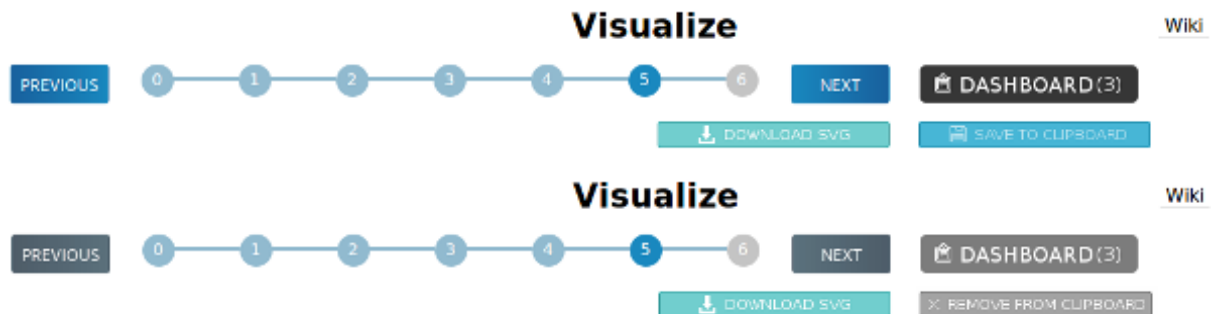


Figure 8 – Dashboard : Boutons externes version 3

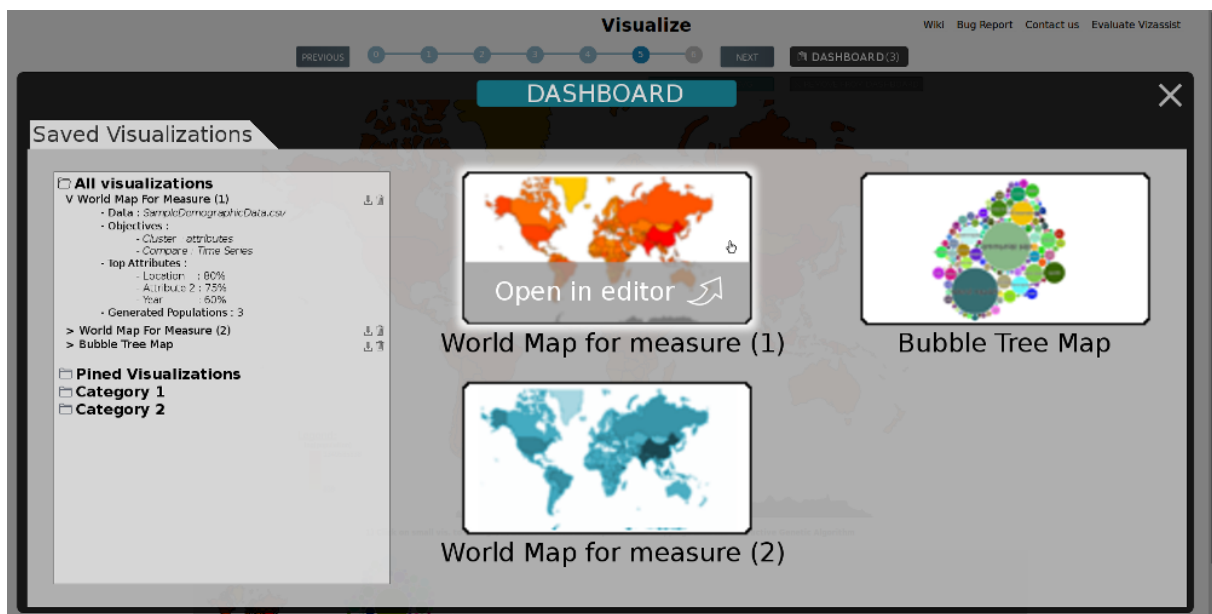


Figure 9 – Dashboard : Première version

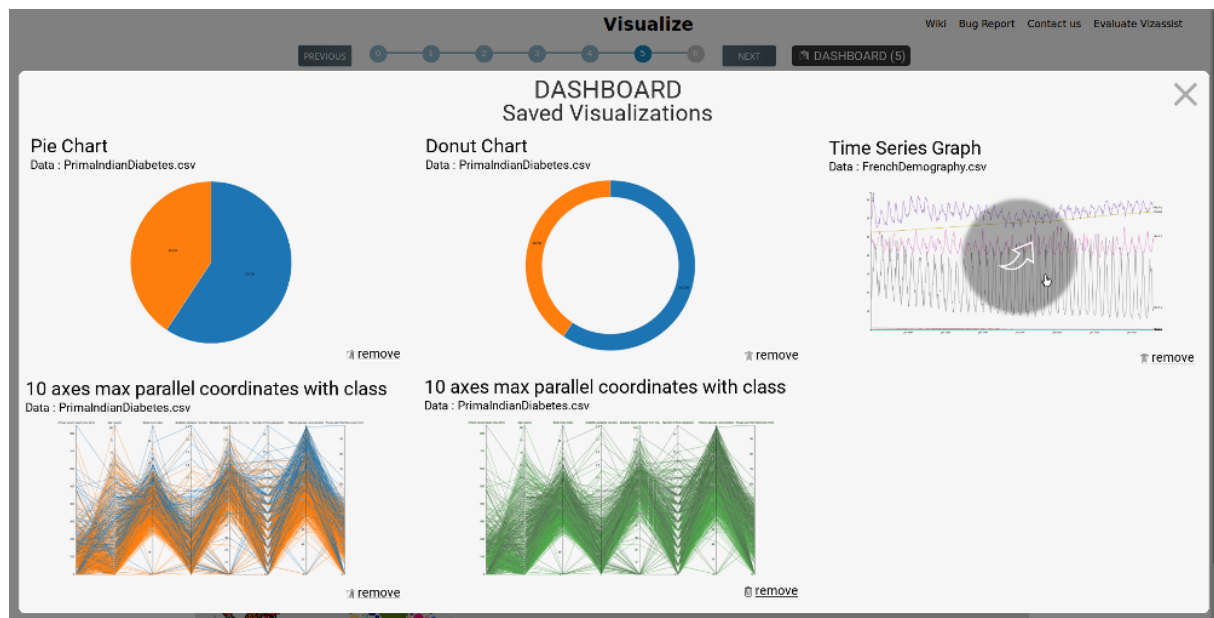


Figure 10 – Dashboard : Version finale

6

Spécifications

1 Fonctionnalités

1.1 F1 - Nouvelle visualisation

L'interface intègre de nombreuses visualisations à l'aide de trois éléments pour chacune d'entre elles :

- Méta-données (identifications, options)
- Paramétrage des objectifs
- Script d'affichage et images miniatures

Pour l'ajout d'une visualisation, il faut donc définir précisément ces trois points. Pour ce projet, la visualisation retenue est le "Donut Chart". Reprenant les principes du camembert classique, cette représentation en cercle en propose une version légèrement différente, apportant de la diversité dans la bibliothèque de la plate-forme.

Script

Le script D3js de base à utiliser est accessible à l'adresse suivante :

<http://bl.ocks.org/dbuezas/9306799>

Afin de correspondre à la plate-forme, il faudra bien entendu adapter ce script. De plus, cette visualisation s'accorde bien avec deux types de regroupements de données : nominal (classes) et ordinal (classes ordonnées). L'ajout de cette visualisation est donc dédoublée, une pour chaque type. Même chose pour les images miniatures permettant leur sélection (**Figure 1**).



Figure 1 – F1 - Miniatures des visualisations Donut Chart nominale et ordinale

Méta-données

Les méta-données des visualisations dans le fichier correspondant (*table-visualisations.csv*) comportent les informations suivantes (Figure 2) :

- L'identification et le nom à leur attribuer
- Le nom des miniatures
- Les attributs : pour chaque portion de données, on a une "proportion" DONUT SIZE et une valeur de poids C qui correspond à la saturation de la couleur. Le premier est numérique et le second ordinal ou nominal.
- L'importance de ces attributs. On récupère les mêmes que pour la visualisation PieChart, qui fonctionne de manière similaire.

ID_VISU	NAME	THUMBNAIL	VISUAL_ATTRIBUTE_NAME	VISUAL_ATTRIBUTE_TYPE	CORRESPONDING_DATA_ATTRIBUTE_TYPE	IMPORTANCE	LIMIT_NB_VALUES
DONUTCHARTORDINAL	Donut Chart for ordinal attribute	donutchartordinal.jpg	DONUTSIZE	size	numeric	90	2000
DONUTCHARTORDINAL	Donut Chart for ordinal attribute	donutchartordinal.jpg	C	color saturation	ordinal	95	20
DONUTCHARTNOMINAL	Donut Chart for class attribute	donutchartnominal.jpg	DONUTSIZE	size	numeric	90	2000
DONUTCHARTNOMINAL	Donut Chart for class attribute	donutchartnominal.jpg	C	color hue	nominal	97.5	10

Figure 2 – F1 - Méta-données associées au "Donut Chart"

Objectifs

Les objectifs couverts par les deux types de cette visualisation, qui correspondent à leur pertinence d'utilisation, sont similaires aux visualisations PieChart et on utilisera de fait les mêmes valeurs dans le fichier des objectifs (*table-objectifs-visualisations.csv*, Figure 3).

Objectives			DONUTCHARTORDINAL	DONUTCHARTORDINAL
Data Mining Tasks >	Analyze >	A class attribute	25	25
	Cluster >	A measure	25	25
	Compare >	Data items	25	25
Properties of visualizations >	Zoom >	Attributes	100	100

Figure 3 – F1 - Objectifs ciblés par "Donut Chart"

1.2 F2 - Visualisation des attributs

Afin de visualiser la distribution des attributs dans les jeux de données, on utilisera la bibliothèque D3js pour créer des diagrammes en barres. Deux types sont à considérer : répartition des classes (nominal) et répartition de valeurs numériques.

Script D3js

Le script D3js de base à utiliser est accessible à l'adresse suivante :

<http://bl.ocks.org/mbostock/3886208>

La figure 4 montre un aperçu du visuel graphique obtenu grâce à ce script. Il sera adapté pour correspondre à la maquette défini dans la section 1 (Chapitre 5) en terme de dimensions. La superposition de plusieurs couleurs dans une même valeur permet une dimension supplémentaire dans le cas d'une répartition multi-classes.

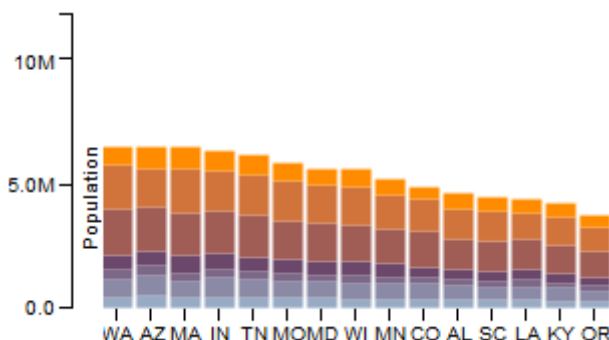


Figure 4 – F2 - Aperçu de la visualisation graphique

Répartition nominale

Dans le cas des classes, la valeur horizontale correspondra à une classe et sa répartition sera représentée par l'axe vertical. Le nombre de barres en largeur ne sera pas limitée et l'échelle de l'axe horizontal doit s'adapter pour correspondre à la largeur du contenant.

Répartition numérique

Dans le cas d'une répartition numérique, le choix est plus délicat et il a été défini que pour garantir une homogénéité dans la fonctionnalité et pour correspondre aux dimensions limitées de la vue, on conservera le diagramme en barre. Cette fois, ce seront des moyennes qui définiront une valeur horizontale pour un nombre maximal de 20 barres. En fonction du résultat, ce nombre peut varier.

Description des fonctions

Cette fonctionnalité se décomposera en plusieurs fonctions :

- **Récupération du type** : Selon le cas nominal ou numérique, le traitement est différent. Cette fonction servira de filtre pour appliquer une répartition dans le diagramme pour l'axe horizontal.
- **Génération des pseudo-classes** : Pour les valeurs numériques graduelles, cette fonction regroupera les valeurs dans des groupes de moyenne sur une échelle proportionnelle de valeur en pourcentage.
- **Création du diagramme** : Pour chaque répartition, cette fonction se servira du script modifié pour générer la vue graphique à l'aide des classes ou pseudo-classes de valeurs.
- **Intégration dans la vue** : Conformément au maquetage, chaque répartition sera intégrée dans la table en étape 3 de l'outil.

1.3 F3 - Tableau de bord

Le tableau de bord, ou Dashboard, se conçoit sur deux niveaux : les fonctions de sauvegarde, de stockage et d'accès, et le panneau d'affichage des visualisations.

Sauvegarde et stockage

Les sauvegardes étant temporaires, liées à une session et l'application sécurisée pour l'utilisateur, les données seront sauvegardées sur le poste local de ce dernier. On utilisera à cet effet les espaces de données *localStorage* grâce au JavaScript. Plus conséquents que les variables de session et de cookies, ce moyen est tout indiqué.

De plus, il a été décidé qu'une instance du tableau de bord est lié à un seul et même jeu de donnée. C'est-à-dire que si l'utilisateur souhaite en changer, il perdra toute sauvegarde effectuée. Cela étant, il n'est pas nécessaire de conserver le jeu de donnée mais uniquement les configurations.

Cette sauvegarde s'effectuera au clic sur le bouton prévu à cet effet au regard des maquettes exposées dans la section 2 (Chapitre 5). Etant donné la nature non critique de l'opération, seul l'aspect du bouton changera, associé d'un court message de confirmation. Un second clic annulera l'action.

L'accès au tableau de bord (Figure 5) pourra se faire depuis n'importe quelle étape, mais la sauvegarde uniquement depuis la dernière. Le chargement d'une sauvegarde ramènera aussi à l'étape 6.

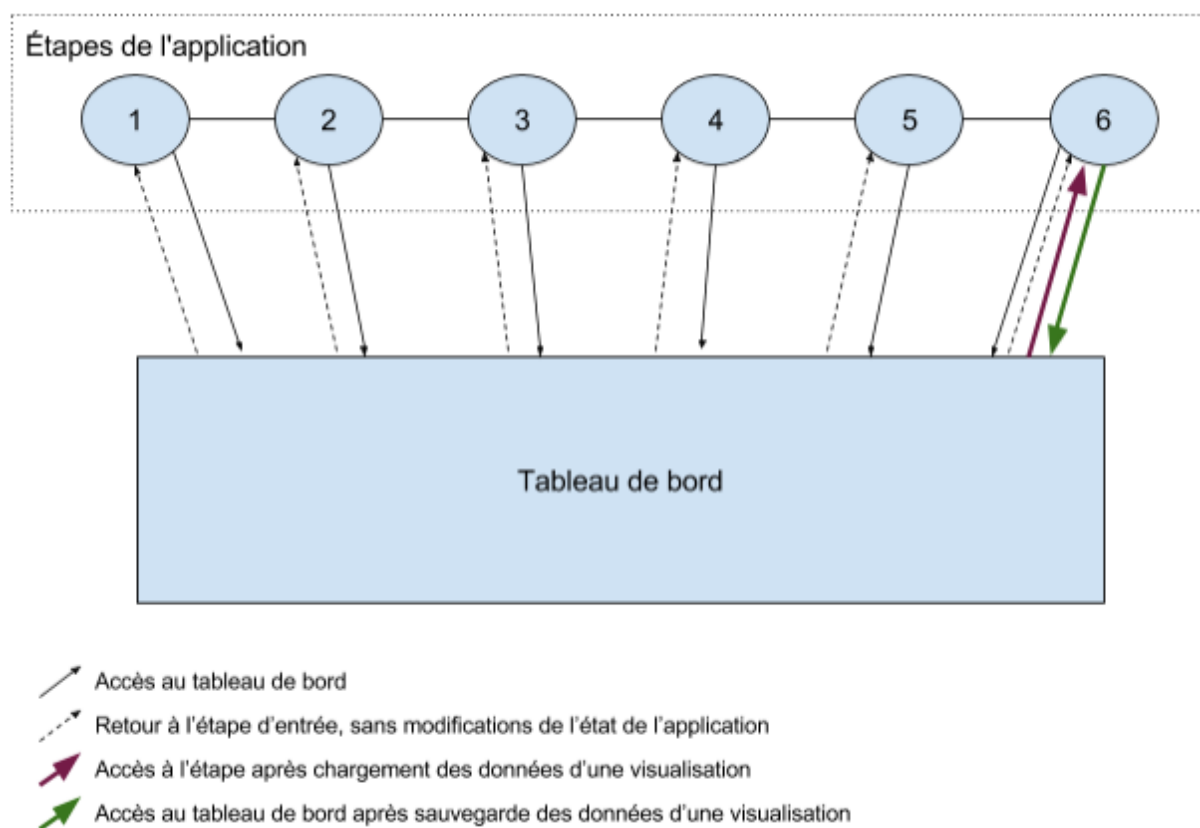


Figure 5 – F3 - Accès et transmissions au tableau de bord

Panneau du tableau de bord

Le panneau du Dashboard est un cadre se superposant au site, dans la même fenêtre. Elle s'ouvre au clic sur le bouton externe "DASHBOARD" et se ferme grâce à un clic en dehors de la fenêtre (dans les marges prévues à cet effet) ou au clic sur une icône de croix dans le coin supérieur-droit.

Ce panneau présente des miniatures des visualisations sauvegardées sous la forme d'une grille. Si le nombre de sauvegarde dépasse le nombre de cases de la grille, une barre de défilement verticale permet de dépasser et d'afficher le reste.

Lors du survol d'une miniature par le curseur de la souris, une icône s'affiche et lors du clic, une bulle de dialogue propose à l'utilisateur de charger la sauvegarde dans l'application au détriment des données actuelles. Il doit donc confirmer son action avant qu'elle prenne effet. Si c'est le cas, le panneau se ferme et l'application se rends à la dernière étape.

Description des fonctions

Voici une brève description des fonctions composant cette fonctionnalité :

- **Sauvegarde d'une configuration** : Récupération des données en cours et ajout dans l'espace *localStorage*.
- **Chargement d'une configuration** : Charge toutes les données de configuration d'une visualisation dans l'instance courante du site et ouvre la page en étape 6.
- **Suppression d'une configuration** : Suppression des données de configuration d'une visualisation depuis le *localStorage*.
- **Réinitialisation des données** : Supprime toutes les configurations présentes dans le *localStorage*. Utile si l'utilisateur souhaite changer de jeu de données ou simplement démarrer une nouvelle session.
- **Affichage des boutons externes** : Génère la vue des boutons externes (accès, sauvegarde, téléchargement) conformément aux maquettes en s'ajoutant au menu de navigation des étapes de l'application.
- **Affichage du Dashboard** : Génère la vue du Dashboard conformément aux maquettes.
- **Visualisation miniature** : Création d'une visualisation miniaturisée pour son affichage dans le Dashboard.
- **Ouverture du panneau** : Déploie le panneau du Dashboard.
- **Fermeture du panneau** : Ferme le panneau du Dashboard.

2 Corrections et améliorations

Les corrections et améliorations à apporter à la plate-forme sont regroupées en 3 catégories. Au cours de cette partie du projet axée sur la recherche, nous avons dégagé mon encadrant et moi-même un grand nombre de problèmes à corriger, tant fonctionnels que visuels. Ces spécifications rendent compte de ceux-ci de manière exhaustive, en se basant sur des tests boîte noire, les tests d'utilisabilité et les observations faites plus couramment à l'utilisation.

Étant donné le nombre de corrections et leur durée de résolution très variable et relativement imprévisible, les efforts seront concentrés sur les plus rapides et sur le développement des fonctionnalités. Certaines améliorations, pour les visualisations notamment, risquent de ne pas aboutir dans le cadre de ce PRD. Elles sont néanmoins toutes référencées et pourront servir dans de futures séquences de corrections.

2.1 Bugs fonctionnels

1. **BF1 - Erreurs PHP**
 - Dû à un conflit de version (antériorité à PHP 5.4), certaines erreurs apparaissent régulièrement.
 - Quelques mots clés sont simplement à supprimer (static).
2. **BF2 - Problème de log**
 - Les fichiers de démonstration sélectionnés ne s'affichent plus dans le fichier de log.
3. **BF3 - Suggestions des visualisations non conformes**
 - "Time series stacked bar chart" n'est proposé pas pour French Demography, mais devrait l'être.
 - Plusieurs visualisations sont proposées pour le jeu de données de démo "VizAssist Project", mais ne devraient pas l'être.
 - Bubble Tree

- Zoomable Tree
 - Zoomable Treemap
 - Zoomable Icicle
 - Cluster Dendogram
4. **BF4** - Time Series Interactive Graph
 - Les valeurs de l'axe Y ne sont pas bonnes.

2.2 Bugs d'affichage

1. **BA1** - Bandeau de la page d'accueil
 - L'image de l'étape 3 est manquante.
2. **BA2** - Texte flottant sur les boutons des étapes
 - Certains ne sont pas conformes aux titres des étapes
3. **BA3** - Texte coupé sur certains axes de certaines visualisations
 - Les visualisations concernées sont :
 - 10 axes max parallel coordinates with nominal attributes
 - 2D scatter plot with color for measure
 - Time series graph
4. **BA4** - Cluster Dendogram : Croisements non désirés
 - Examiner la possibilité de les éviter.
5. **BA5** - Box Plot for nominal attribute : Labels
 - L'affichage des labels n'est pas correct et doit être légèrement corrigé de manière à obtenir le format suivant :
 - Class : [NomClasse]

Plusieurs problèmes concernant la légende n'ont pas été référencés ici car une amélioration est prévue pour celle-ci dans la sous-section suivante.

2.3 Améliorations de l'UI et des visualisations

Interface Utilisateur

1. **Am1** - Textes d'aide
 - Ajouter des textes d'aide à la navigation en entête des étapes posant des problèmes de compréhension et mettre plus en avant les existants.
 - Étapes concernées : 2, 3, 4 et 6.
2. **Am2** - Sélection par défaut à l'étape 2
 - Faire en sorte que l'option "I have no idea, se please make me some suggestions" soit sélectionnée par défaut.
3. **Am3** - Sélection multiple en étape 4
 - Déplacer les boutons "Select All" et "Unselect All" en haut de la liste en étape 4.
4. **Am4** - Bouton EVALUATE
 - Déplacer le bouton d'évaluation présent en fin d'étape 6 et le changer de couleur.
5. **Am5** - Titre des sections
 - Tenter d'améliorer leur visibilité.
6. **Am6** - Affichage de la légende
 - Pour les visualisations possédant une légende :
 - Largeur flexible évitant les retours chariots intempestifs.
 - Mettre en valeur la différence entre un label et sa valeur.
7. **Am7** - Nom des visualisation en étape 5
 - Afficher le nom des visualisations dans le bandeau de sélection.
8. **Am8** - Nom du jeu de données
 - Afficher le nom du jeu de donnée, s'il existe, dans l'interface au cours des différentes étapes.
9. **Am9** - Dispositions en étape 1
 - Ré-organiser la disposition des éléments de manière à optimiser l'espace.

Visualisations

1. **Av1** - Bubble Tree et Zoomable Treemap
 - Faire en sorte que la taille des éléments (cercles, blocs) dépende de leur valeur.
2. **Av2** - (Un)Directed Graph force-directed layout
 - Séparer davantage les noeuds pour mieux visualiser les données.
3. **Av3** - Multiline chart with brushing
 - Colorer le nom des séries en bleu et rouge pour mieux les différencier.
4. **Av4** - Small multiples time series
 - Afficher le nom des séries sur le dessus de la visualisation.

7

Organisation

1 Tâches et planning

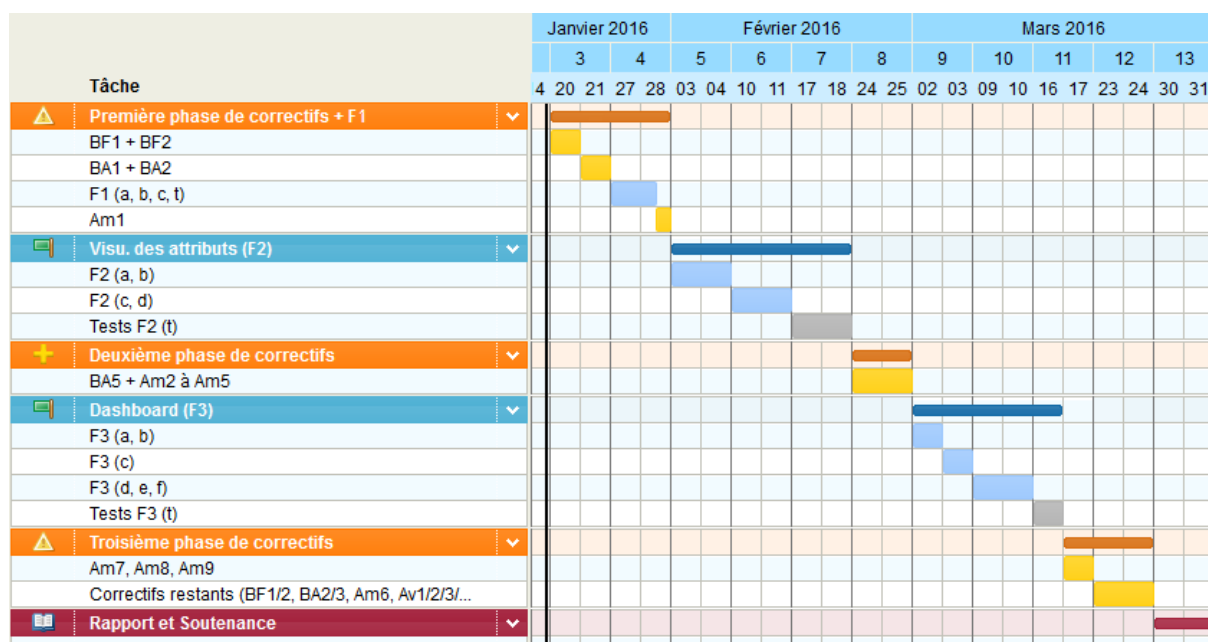


Figure 1 – Planning prévisionnel des phases de développement

Les différentes tâches à réaliser pour la partie développement se divisent en deux catégories :

- Les correctifs et améliorations
- Le développement des fonctionnalités

Correctifs

La division en tâche des correctifs se fait naturellement : une correction ou une amélioration est une tâche en soi. Voici un rappel des identifications pour chacune d'entre elles :

- **BF1** - Erreurs PHP
- **BF2** - Problème de log
- **BF3** - Suggestions des visualisations non conformes

- **BF4** - Time Series Interactive Graph
- **BA1** - Bandeau de la page d'accueil
- **BA2** - Texte flottant sur les boutons des étapes
- **BA3** - Texte coupé sur certains axes de certaines visualisations
- **BA4** - Cluster Dendogram : Croisements non désirés
- **BA5** - Box Plot for nominal attribute : Labels
- **Am1** - Textes d'aide
- **Am2** - Sélection par défaut à l'étape 2
- **Am3** - Sélection multiple en étape 4
- **Am4** - Bouton EVALUATE
- **Am5** - Titre des sections
- **Am6** - Affichage de la légende
- **Am7** - Nom des visualisation en étape 5
- **Am8** - Nom du jeu de données
- **Am9** - Dispositions en étape 1
- **Av1** - Bubble Tree et Zoomable Treemap
- **Av2** - (Un)Directed Graph force-directed layout
- **Av3** - Multiline chart with brushing
- **Av4** - Small multiples time series

Comme précisé dans le cahier de spécification, un certain nombre de ces problèmes ne pourra être corrigé dans les délais impartis. Ainsi, le planning est prévu pour apporter une solution aux plus importants en premier, puis lors des différentes phases de correctifs jusqu'à la dernière où un maximum sera fait pour corriger les manquantes.

Fonctionnalités

Les trois fonctionnalités F1, F2 et F3 peuvent se décomposer en plusieurs tâches, au regard des fonctions définies dans le cahier des spécifications.

- **F1 - Nouvelle visualisation** : Intégré dans la première phase de correctifs, cette fonctionnalité ne comporte pas de défi majeur et sera effectué très rapidement.
 - (a) Insertion des informations dans les fichiers de méta-données et d'objectifs
 - (b) Adaptation du script D3js
 - (c) Tests et corrections
- **F2 - Visualisation des attributs** : Deux semaines sont prévus pour ces tâches. Une phase de test aura lieu au cours de la pause pédagogique de février.
 - (a) Adaptation du script D3js
 - (b) Vérification des types de données et regroupements en pseudo-classes des attributs numériques
 - (c) Intégration
 - (d) Tests et corrections
- **F3 - Tableau de bord** : La dernière partie du planning est dédié à cette fonctionnalité majeure. Une phase de test est prévue avant les derniers correctifs.
 - (a) Ajout des boutons externes dans l'UI
 - (b) Gestion de l'ouverture du Dashboard et création de la vue
 - (c) Fonctions de sauvegarde en localStorage
 - (d) Intégration des fonctions précédentes dans l'UI
 - (e) Affichage des visualisations sauvegardées dans le Dashboard
 - (f) Chargement d'une visualisation
 - (g) Tests et corrections

2 Méthodologies et livrables

Méthodologies

Une méthodologie agile sera employée pour la partie développement du projet. Comme le montre le planning prévisionnel dans la section précédente, plusieurs phases sont prévues : axées alternativement sur les correctifs et sur le développement de fonctionnalité. Les premières permettront, en plus de corriger les bugs et effectuer les améliorations mineures, de tester et valider les secondes. Le découpage en très petites tâches permet aussi ce développement progressif et où chacun a un impact minimal sur l'ensemble.

La gestion des versions se faisant avec un dépôt Git, une branche a été créée pour ce PRD. Dans le processus de développement et de validation, je procéderais de cette manière :

- Commit : La plus petite modification. Idéalement, une tâche ou une fonction.
- Push : Un groupe de correctifs ou une fonctionnalité.

Livrables

Sont attendus pour la validation de ce projet plusieurs éléments :

- Maquettes et Spécifications
- Branche sur le dépôt contenant toutes les mises à jours
- Une instance du site web sur laquelle sera exécutée la version modifiée de la branche.
- Rapport sur les bugs restants
- Rapport final

Ce rapport contient les maquettes et les spécifications. La version finale de celui-ci, à livrer pour la fin du projet, contiendra en plus les détails quant au développement et travaux effectués sur la branche du dépôt git. Celle-ci pourra être fusionnée sur la branche maître du serveur de production, à la demande de l'encadrant. Dans tous les cas, une instance parallèle à l'application publique sera mise à jour régulièrement avec les dernières modifications apportées.

Les bugs et autres problèmes non résolus ou apportés par les modifications effectuées et qui auront été identifiés feront l'objet d'un rapport particulier à l'attention des prochains collaborateurs à VizAssist.

Troisième partie

Développement

Dans cette troisième et dernière partie, nous aborderons toute la mise en oeuvre du projet, réalisée au cours du semestre 10 à la suite des parties précédentes liées à la recherche. Nous détaillerons tout d'abord les travaux effectués au regard des spécifications, suivi de la description de la campagne de test associée. Nous terminerons par une section tirant le bilan de ces activités.

8

Mise en oeuvre

1 F1 - Visualisation Donut Chart

Lors de la première partie du développement, en parallèle de plusieurs correctifs, j'ai intégré la visualisation Donut Chart à la bibliothèque. Cet ajout a été rapidement mis en place grâce aux spécifications détaillées. Pour rappel, 3 éléments étaient à intégrer :

- Les méta-données
- Le paramétrage des objectifs
- Le script d3js

Pour les deux premiers points, les valeurs ont été ajoutées aux fichiers comme spécifié dans la partie 1.1 (Chapitre 6) . En ce qui concerne le script d3js, il a fallu l'adapter pour qu'il s'intègre correctement dans le projet. Pour cela, je me suis basé sur le script existant pour la visualisation "PieChart". En effet, ces deux visualisations sont très similaires.

Voici un extrait de code montrant la principale différence entre ces deux visualisations : le rayon interne du "donut" (innerRadius).

```
1 var arc = d3.svg.arc()  
2   .outerRadius(radius - 10)  
3   .innerRadius(Math.max(radius - 70, radius*70/100));
```

Il a fallu également créer deux scripts différents : **donutchartnominal.js** et **donutchartordinal.js** pour traiter respectivement les types de données nominales et ordinales. Bien qu'étant identiques, les deux scripts doivent être présent afin que le programme reconnaisse la possibilité à la visualisation de traiter les deux cas. La seule différence, mise à part les conventions de nomage, concerne l'aspect graphique en proposant deux colorations distinctes en se basant sur des échelles de couleurs propres à d3js :

- Échelle de valeur de catégorie 10 pour la version nominale : `d3.scale.category10()`
- Échelle de valeur de catégorie 20 pour la version ordinale : `d3.scale.category20()`

La figure 1 présente ce que donne la visualisation dans son contexte, une fois intégrée à la bibliothèque.

2 F2 - Visualisation des attributs

Cette seconde fonctionnalité aura prit beaucoup plus de temps que prévu sur le planning. Cela n'a cependant pas été perdu dans la mesure où elle a bénéficié d'un travail plus important, d'une intégration plus complète et de beaucoup d'attention aux détails.

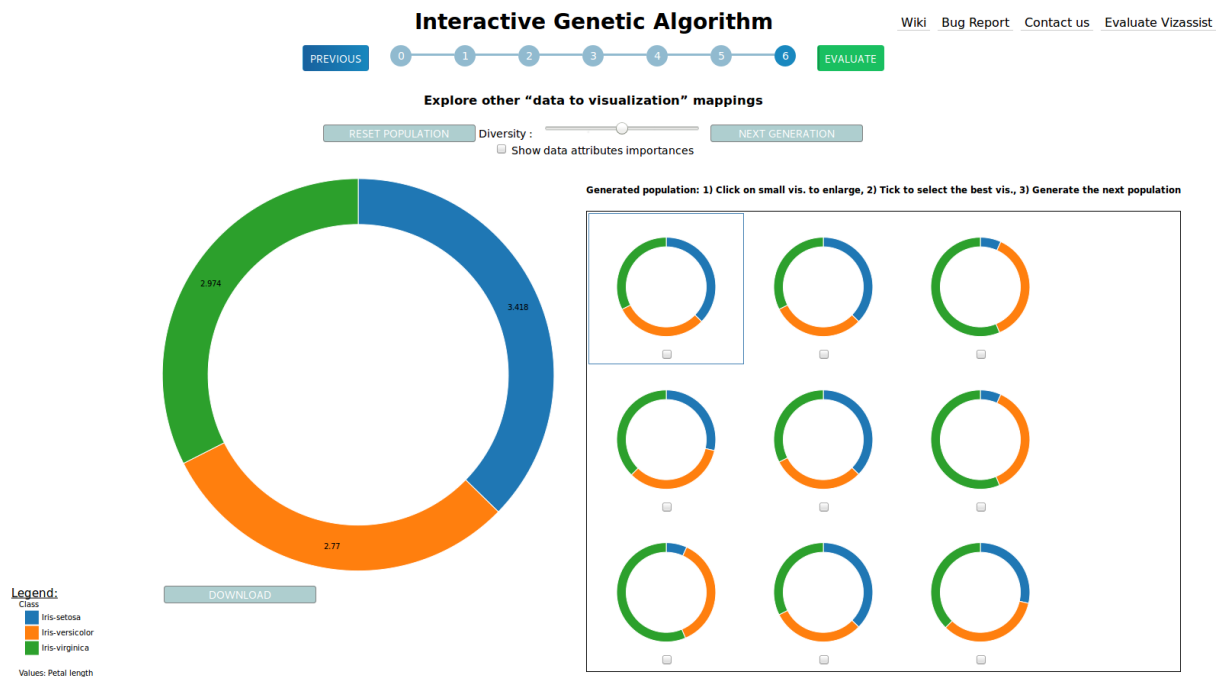


Figure 1 – F1 - Visualisation "Donut Chart" dans l'étape 6 de l'application.

2.1 Histogramme d3js

Tout d'abord, il était prévu d'implémenter la fonctionnalité sous la forme d'histogrammes représentant la répartition des valeurs pour chaque attributs d'un jeu de données. Pour cela, les spécifications prévoyaient d'intégrer un script d3js trouvé dans la bibliothèque publique du site officiel. Cependant, lors de la première séance de travail sur celui-ci, je me suis vite aperçu qu'il était inadapté pour plusieurs raisons : taille trop importante, présence d'une légende, d'axes, de plusieurs niveaux et couleurs, etc. En somme, le script était bien trop complexe et intégrait trop de fonctions alors que l'implémentation que nous souhaitions se voulait la plus simple et réduite possible, pour s'intégrer dans l'interface utilisateur existante.

C'est pourquoi j'ai préféré laisser de côté ce script existant et repartir de zéro en codant moi-même un histogramme simplifié et répondant précisément au besoin.



Figure 2 – F2 - Histogramme généré en d3js

Voici le code d3js générant l'histogramme présenté en figure 2 :

```
1 // Dimensions
2 var chartWidth = 300, // Width of the diagram
3     chartHeight = 50, // Height of the diagram
4 // SVG container
5 var SVGcontainer = visuTd.append("svg")
6     .attr("class", "repartitionChart")
7     .attr("width", chartWidth)
8     .attr("height", chartHeight);
9 // Valued bar
10 SVGcontainer.selectAll(".rectValued")
11     .data(d3data)
```

```

12 .enter().append("rect")
13 .attr("id", function(d, i) {
14     return "rectValued_" + i;
15 })
16 .attr("width", barWidth)
17 .attr("height", function(d) {
18     return yRange(d.val);
19 })
20 .attr("fill", function(d){return d.color;})
21 .attr("y", function(d) {
22     return (d.val == 0) ? chartHeight - 1 : chartHeight - yRange(d.val);
23 })
24 .attr("x", function(d, i) {
25     return (this.togglevalues) ? xRange(i) : barWidth * i;
26 })

```

Quelques variables sont définies en amont :

- visuTd est le conteneur de l'histogramme, c'est à dire la cellule de la table présente dans la vue.
- d3data contient les données que l'ont souhaite visualiser, c'est à dire les valeurs de l'attribut.

On retrouve également **yRange**, qui correspond à une échelle de valeur allant de 0 à la hauteur de l'histogramme et sert à définir la hauteur de chaque barre en fonction de la valeur donnée.

Échelles de données

La bibliothèque D3js propose une fonction très utile pour la visualisation de données : les **échelles** appelées **scale**. Grâce à ce système, on peut définir une intervalle dans laquelle dimensionner les données d'entrée. On l'utilise généralement pour afficher des coordonnées rapportées à une échelle appropriée à l'écran. La figure 3 illustre son fonctionnement.

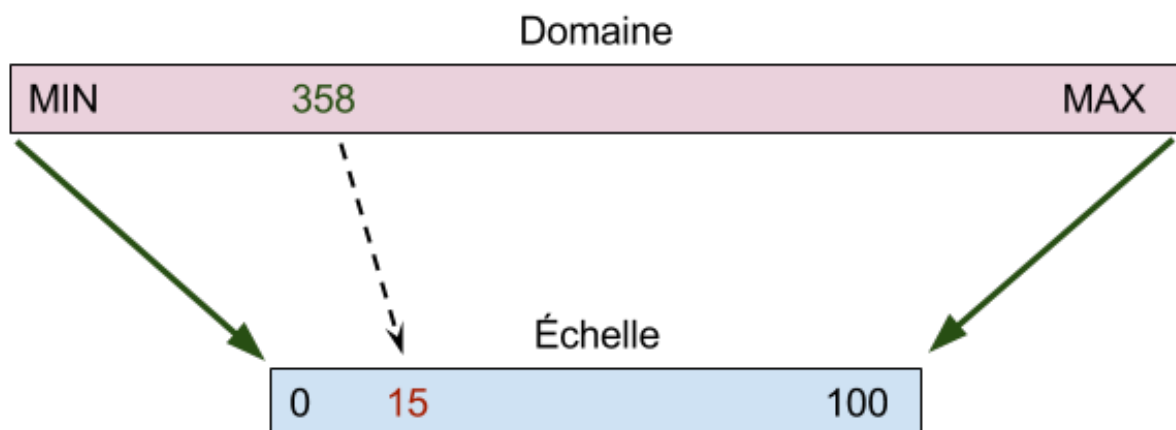


Figure 3 – Illustration du système d'échelle de d3js.

```

1 var yRange = d3.scale.linear()
2   .domain([0, maxMappedVal])
3   .range([1, chartHeight])

```

Dans notre cas, cela est utilisé pour la hauteur des barres de l'histogramme. L'échelle de hauteur va donc aller de 1 à la hauteur maximum de l'histogramme, et le domaine des données ira de 0 à la valeur maximale existante. Dans l'extrait de code ci-dessus, celle-ci est représenté par la variable **maxMappedVal**. La déclaration de cette échelle contient aussi la fonction **linear()** qui permet de préciser que le type de mise à l'échelle doit être linéaire.

Une fois l'échelle définie, il suffit de l'appeler en lui passant la valeur en paramètre et celle-ci retourne alors la nouvelle valeur échelonnée : **yRange(d.val)** où d.val est la donnée d'origine.

Ce système est aussi utilisé pour la position horizontale des barres. En effet, en d3js, le positionnement des éléments graphiques se fait de manière absolue ou relative à leur conteneur, grâce aux propriétés CSS standards. Le domaine va donc de 0 au nombre de barres désiré tandis que l'échelle va de 0 à la position la plus avancée (c'est-à-dire la largeur de l'histogramme moins la largeur d'une barre).

En ayant l'indice de la valeur dans le jeu de données, on peut alors déterminer l'écart horizontal entre le bord gauche de l'histogramme et la barre verticale qui la représente. L'extrait de code suivant considère le nombre de barres désiré comme étant le nombre d'éléments : `localDataset[k].values.length`, où `localDataset` est une copie temporaire locale des données.

```
1 xRange = d3.scale.linear()
2   .domain([0, localDataset[k].values.length])
3   .range([0, chartWidth - barWidth]);
```

2.2 Répartitions et valeurs

Une fois le script détaillé, il a fallu l'implémenter dans le module. Pour le développer, j'ai pu utiliser un simple jeu de données de test créé pour l'occasion. Mais l'objectif est bien de s'en servir pour visualiser la répartition des attributs. Dans le module RequireJS nommé **Variables**, j'ai ainsi dû adapter mon script en changeant les données d'entrée et intégrer l'histogramme dans la table de l'UI.

Visualisation des valeurs numériques

La structure étant relativement complexe, j'ai au départ eu des difficultés à conceptualiser correctement ce qu'une répartition voulait signifier. C'est pourquoi j'ai d'abord visualisé les données brutes plutôt que leur répartition. La figure 4 montre à quoi ressemble cette visualisation.

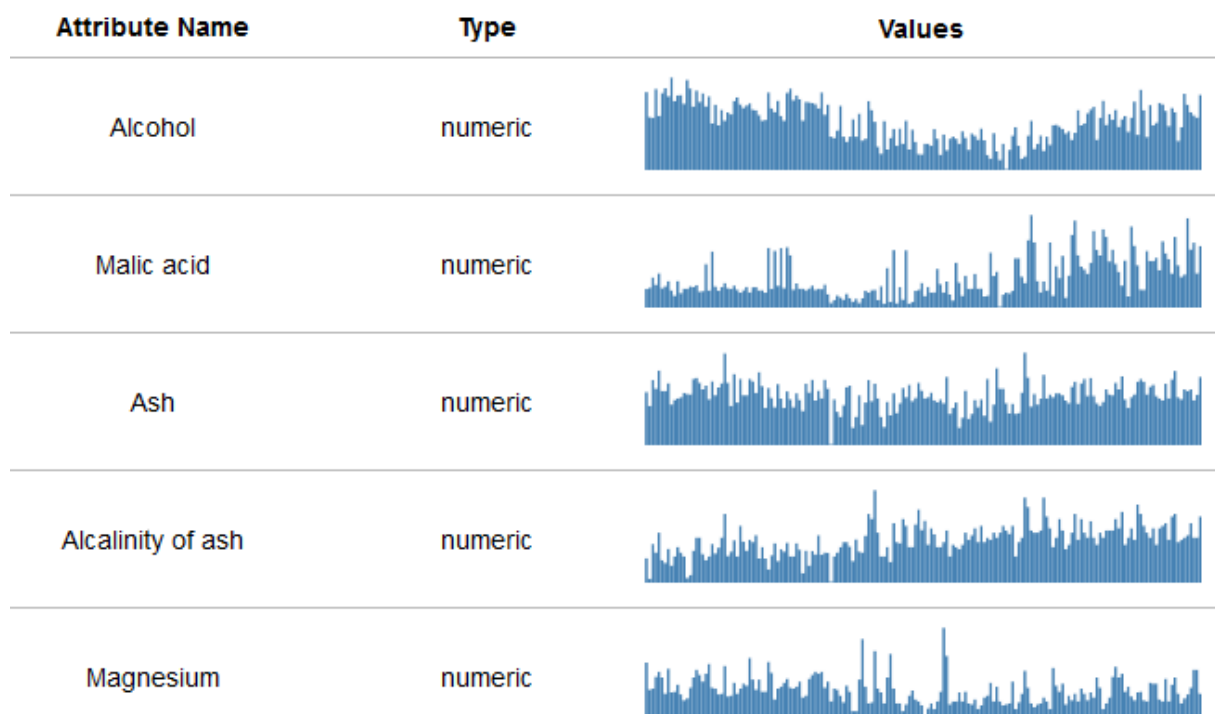


Figure 4 – F2 - Première visualisation des données numériques.

Ce n'était pas l'objectif initial, mais l'idée semblait intéressante pour M. Venturini. J'ai donc poursuivi en direction de la répartition réelle des valeurs. Mais pour cela, il était nécessaire de distinguer deux cas bien précis : les valeurs numériques et les autres.

Répartition des valeurs numériques

```

1  if (localDataset[k].type == "numeric") {
2      // 1) Parsing values into float to prevent errors due to data saved in strings
3      values = values.map(function(d) {
4          return parseFloat(d);
5      })
6      // 2) Ascending data sort
7      values.sort(function(a, b) {
8          return a - b;
9      });
10     var steps = 30,
11         maxEpsilon = (d3.max(values) - d3.min(values)) * 0.01, // Scale max bound error ←
12         // prevention
13         stepRange = d3.scale.linear()
14         .domain([d3.min(values), d3.max(values) + maxEpsilon])
15         .range([0, steps]);
16     for (var i = 0; i < steps; i++) {
17         // Mouseover interval displays (for future purposes)
18         var delta = (d3.max(values) - d3.min(values)) / steps,
19             intervalInf = parseFloat(d3.min(values) + i * delta).toFixed(2),
20             intervalSup = parseFloat(d3.min(values) + (i + 1) * delta).toFixed(2);
21         // 3) d3data initialisation
22         d3data[i] = {
23             val: 0,
24             hover: "[" + intervalInf + " - " + intervalSup + " [",
25             color: barColor
26         };
27         // 4) Values incrementation
28         for (var i in values)
29             d3data[parseInt(stepRange(values[i]))].val++;
30     }

```

Dans cet extrait de code, les données de l'attribut courant `k` sont contenues dans l'objet `localDataset[k]`. Pour obtenir une répartition, j'ai procédé de la manière suivante :

1. Parsing des données en float. Les fonctions mathématiques peuvent retourner des erreurs car le typage est dynamique et les nombres peuvent être stockés en chaînes de caractères.
2. Tri croissant des données.
3. Définition d'une échelle de 0 au nombre de barres voulues, avec pour domaine les données minimales et maximales (+ un epsilon).
4. Initialisation d'un tableau de données d'affichage à 0. Celles-ci correspondent aux valeurs des barres de l'histogramme, il y en a donc un nombre égal au nombre de barres.
5. Pour chaque valeur de l'attribut :
 - (a) Récupération de la valeur mise à l'échelle.
 - (b) Incrémentation de la valeur dans les données d'affichage à l'index égal à la valeur de l'attribut mise à l'échelle.

Cette procédure permet d'obtenir un tableau de `steps = 30` valeurs où chacune d'elle correspond au nombre d'occurrences des valeurs de l'attribut de l'intervalle une fois mis à l'échelle. Par exemple, pour un domaine allant de 0 à 30, si on a 3 valeurs comprises dans `[2.0 - 3.0[` alors `d3data[2]` sera égal à 3.

Répartition des valeurs non-numériques

On va appliquer le même principe pour les données de type non numériques (noms, classes, lieux, url, ...) mais cette fois en comptant le nombre d'occurrences non plus dans une échelle de données mais de manière absolue. Il serait absurde de vouloir obtenir 30 barres dans un histogramme ne contenant que 5 valeurs possibles. De la même manière, le tableau de données d'affichage va contenir le nombre d'occurrences pour chaque valeur :

1. Tri alphanumérique des valeurs
2. Initialisation de la première valeur du tableau d'affichage à 1.
3. Pour chaque valeur de l'attribut suivante :
 - Si la valeur est la même que la précédente, on incrémente la valeur courante du tableau.
 - Sinon, on incrémente l'index du tableau et on initialise sa valeur à 0.

Visualisation des valeurs non numériques

En ce qui concerne la visualisation des valeurs, conservée en parallèle de la répartition, il restait l'affichage des données non numériques. Pour cette dernière, visualiser des noms et des url grâce à une hauteur de barre n'était pas très intuitif. Après réflexion, nous avons eu l'idée de simplement afficher toutes les barres de la même hauteur, mais de couleurs différentes.

Pour cela, une option est rajoutée dans le tableau des données d'affichage, le transformant en objet à plusieurs attributs : la valeur et la couleur.

Types d'histogrammes

A terme, il existe 4 visualisations possibles :

- Valeurs numériques
- Valeurs non numériques
- Répartition de valeurs numériques
- Répartition de valeurs non numériques

Grâce au modèle d'objet JavaScript, chaque cas est traité en transformant les données d'entrées en une nouvelle variable contenant les valeurs qui seront affichées dans l'histogramme. Ainsi, l'objet final `d3data` est inséré en paramètre dans celui-ci. Cet objet est un tableau de taille variable contenant des objets ayant chacun trois attributs : une valeur, une couleur et un texte. Ce dernier ayant une utilité décrite dans la section suivante ([subsection 2.3](#)).

```

1 d3data = [
2   {
3     val : "Nombre réel",
4     color : "Couleur hexadécimale",
5     hover : "Texte"
6   }
7 ]

```

2.3 Info-bulles

Afin d'améliorer la visualisation, et pour éviter l'ajout d'une légende, une info-bulle a été rajouté au survol de la souris sur les barres de l'histogramme. Cette fonction supplémentaire a été intégrée vers le milieu du développement, constatant la lecture difficile pour certaines données de grande taille.

Le texte de l'info-bulle peut contenir deux choses : soit la valeur précise dans le cas de la visualisation des données, soit l'intervalle que représente la barre pour une distribution, ainsi que le nombre de valeurs comprises dans celle-ci. C'est pourquoi cette information est rajoutée dans la structure de l'objet `d3data`, comme décrite dans la section précédente, lors de la définition des valeurs pour chaque type de données.

```

1 SVGcontainer.selectAll(".rectValued")
2   .data(d3data)n
3   .enter().append("rect")
4   // [...] Reste du script générant l'histogramme [...]
5   // Evènement d'entrée dans la barre
6   .on("mouseover", function(d, i) {
7     // Apparition, contenu et positionnement de l'info-bulle
8     var tooltipText = (that.togglevalues) ? "Value: " + d.hover : d.hover + "<br ↩️"
9       + ">0occurrence(s): " + d.val;
10    tooltip.transition()
11      .duration(200)

```

```

11     .style("opacity", .9);
12     tooltip.html(tooltipText);
13     d3.select(this.parentNode).select("#rectBlank_" + i).attr("fill", blkBarColor_hover);
14     d3.select(this).style("opacity", "0.5");
15 }
16 // Evènement de mouvement dans la barre
17 .on("mousemove", function() {
18     // Déplacement de l'info-bulle relativement à la position du curseur
19     tooltip.style("left", (d3.event.pageX + 20) + "px")
20     .style("top", (d3.event.pageY - 50) + "px");
21 })
22 // Evènement de sortie
23 .on("mouseout", function(d, i) {
24     // Disparition de l'info-bulle
25     tooltip.transition()
26         .duration(500)
27         .style("opacity", 0);
28     d3.select(this.parentNode).select("#rectBlank_" + i).attr("fill", blkBarColor);
29     d3.select(this).style("opacity", "1");
30 });

```

L'info-bulle en elle-même est générée lors de la création de l'histogramme, en d3js. En réalité, une petite <div> est ajoutée pour chaque barre, avec une position relative à la position de la souris. Les événements suivants manipulent l'info-bulle associée : lorsque le curseur entre dans la barre (**mouseover**), elle disparaît, lorsqu'il se déplace (**mousemove**), elle le suit, et lorsqu'il quitte la zone, (**mouseout**) elle disparaît.

Pour simplifier son affichage, j'ai dédoublé le script de l'histogramme en inversant les barres afin de remplir en hauteur l'histogramme. Cela permet de mieux visualiser les barres très petites, lors de grands écarts entre les occurrences ou valeurs. Cette méthode est représentée par la figure 5.

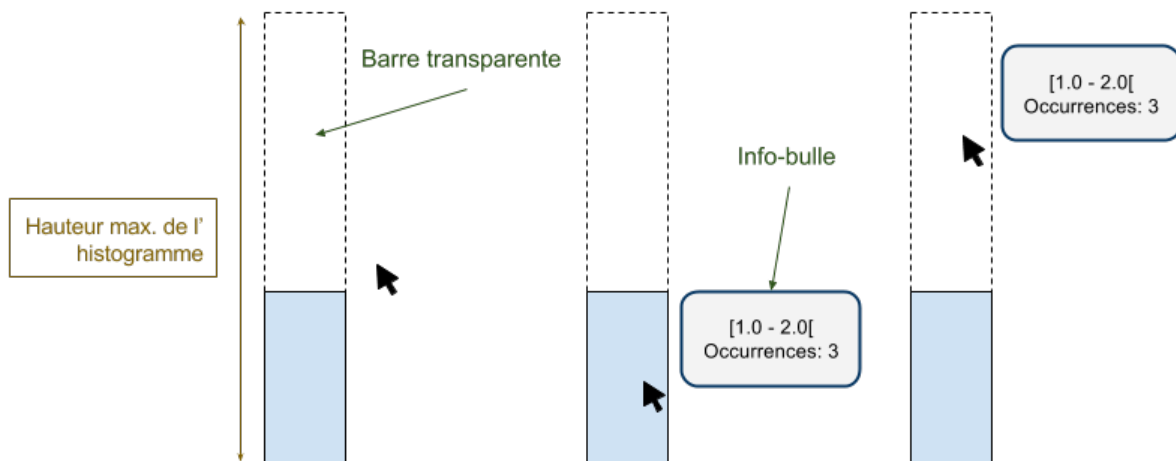


Figure 5 – F2 - Affichage de l'info-bulle selon l'évènement lié au curseur.

2.4 Tri

Comme précisé précédemment, il a été décidé de conserver la visualisation de la valeur des attributs en parallèle de leur répartition. Cependant, pour rendre cette fonction réellement utile, j'ai rajouté un système de tri par attribut. Ce tri (croissant ou décroissant, numérique ou alphanumérique) impacte alors toutes les autres visualisations permettant de comparer les valeurs entre les attributs.

Pour cela, il a fallu rajouter plusieurs choses :

1. Une variable de tri sur le module des Variables, contenant l'index de l'attribut, ainsi qu'un booléen définissant l'ordre de tri.

2. Une copie locale des données, pour ne pas impacter l'ordre de celles-ci dans les autres modules.
3. Des fonctions de tri s'appliquant sur un attribut et impactant tous les autres.
4. Des boutons avec leur événements de tri associés.

Variables de tri

Ce point est important car les variables ont dans un premier temps été déclarées localement. Cela créait des erreurs lors des changements de modules et de données. En effet, les valeurs de ces dernières n'étaient pas réinitialisées étant donné que le module n'est pas supprimé puis rechargé mais simplement réinitialisé. Il fallait donc modifier cela pour transformer ces variables en **attributs du module**, puis ajouter une règle de ré-initialisation dans la fonction `reset()` prévue.

```

1  /**
2   * Reset user input data importance
3   * @return {type}
4   */
5  this.reset = function reset() {
6    if (this.varDefined){
7      this.togglevalues = false;
8      this.sortedby = 0;
9      this.sortedReverse = false;
10     this.varDefined = false;
11   }
12 };

```

Dans cet extrait de code montrant cette fonction de ré-initialisation, on peut retrouver plusieurs attributs dont `this.varDefined` : celui-ci sert à vérifier si le module est chargé ou non. S'il ne l'est pas, cet attribut ne l'est pas non plus et retourne faux. Ainsi, en la passant sur vrai lors de l'initialisation du module, on sait que les données sont chargées et en retournant sur le module par la suite, la fonction `reset()` est appelée et tous les attributs sont redéfinis par défaut.

2.5 Interface Utilisateur

Enfin, la dernière étape est d'intégrer la fonctionnalités et toutes ses fonctions dans l'interface utilisateur existante. Pour cela, j'ai modifié la structure de la table des attributs :

- Fusion des colonnes de poids et du curseur de sélection
- Ajout de la colonne pour la visualisation
- Corrections visuelles mineures (contours, indications ...)
- Ajout d'un bouton pour alterner entre les deux vues (répartition et valeurs)
- Ajout des icônes de tri pour les visualisations des valeurs

Les figures suivantes présentent cette intégration visuelle : la figure 6 montre l'état de la table avant toute implémentation, puis les figures 7 et 8 montrent respectivement ce à quoi peut ressembler la visualisation de la répartition et des valeurs pour les attributs. Sur ces images, la base de test *Iris* a été utilisée.

3 Correctifs et améliorations diverses

En plus des fonctionnalités, de nombreux correctifs et améliorations étaient prévus. Seul une fraction d'entre eux ont pu être réalisés dans les temps, dont voici la liste :

- **BF1** - Erreurs PHP
- **BF2** - Problème de log
- **BA1** - Bandeau de la page d'accueil
- **BA2** - Texte flottant sur les boutons des étapes
- **Am1** - Textes d'aide
- **Am2** - Sélection par défaut à l'étape 2
- **Am3** - Sélection multiple en étape 4

Select your attributes

PREVIOUS 0 1 2 3 4 5 6 NEXT

If you think that some attributes are more important than others for your visualizations, then please let VizAssist know about it. If all attributes are equally important, then just click next.

Attribute Name	Type	Importance		
Petal width	numeric	50	Useless	Mandatory
Petal length	numeric	50	Useless	Mandatory
Sepal length	numeric	50	Useless	Mandatory
Sepal width	numeric	50	Useless	Mandatory
Class	nominal	50	Useless	Mandatory

Figure 6 – F2 - Page des attributs avant l'implémentation de la visualisation.

Select your attributes

PREVIOUS 0 1 2 3 4 5 6 NEXT

If you think that some attributes are more important than others for your visualizations, then please let VizAssist know about it. If all attributes are equally important, then just click next.

Attribute Name	Type	Distribution $\neq$ Values	Importance (-) Useless / Mandatory (+)
Petal width	numeric		50
Petal length	numeric		50
Sepal length	numeric		50
Sepal width	numeric		50
Class	nominal		50

Figure 7 – F2 - Page des attributs avec visualisation de la distribution.

— Am4 - Bouton EVALUATE

D'une manière générale, les corrections liées aux visualisations particulières n'ont pas été traitées, préférant me concentrer sur les problèmes d'interface et au dérogage le plus complet possible des ajouts fonctionnels.

BF1 - Erreurs PHP

Cette correction a été effectuée au tout départ du projet, afin de pouvoir faire fonctionner le serveur PHP local. En effet, de nombreuses fonctions n'étaient pas déclarées **static** alors qu'elle étaient appelées de manière statique dans tout le programme. Or, si cela était possible avant PHP 5.4, cela est depuis devenu obsolète.

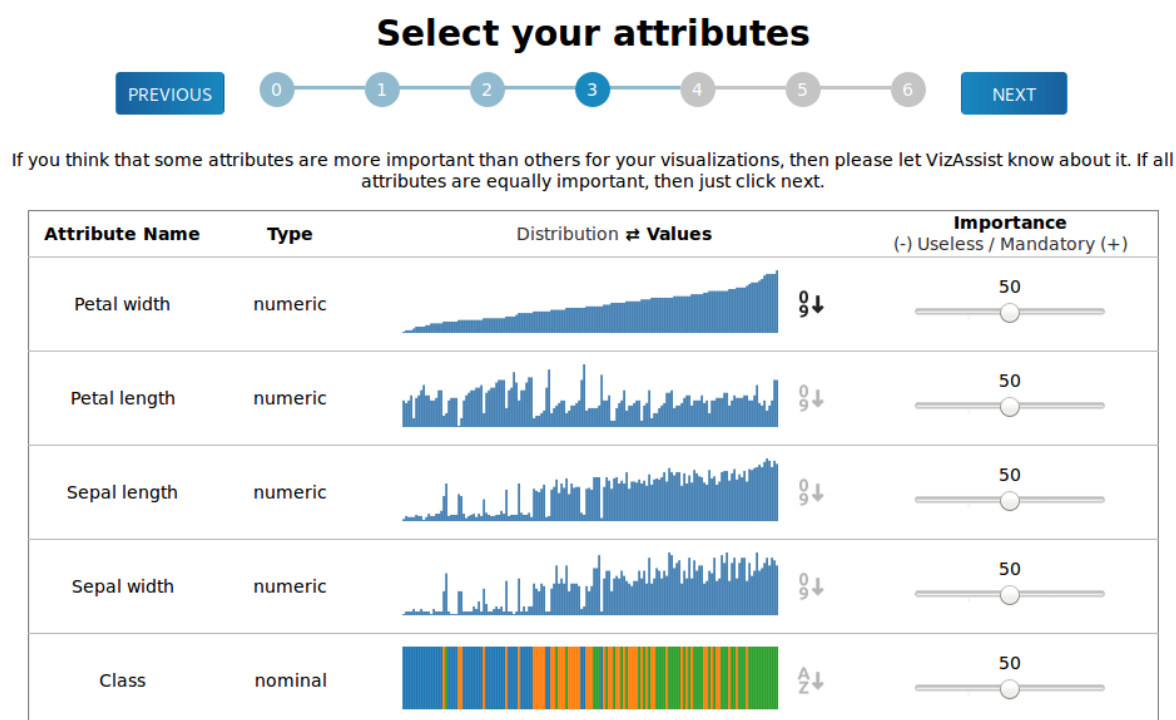


Figure 8 – F2 - Page des attributs avec visualisation des valeurs, triées selon le premier attribut.

BF2 - Problème de log

Une erreur dans le code empêchait de sauvegarder le nom du fichier de démonstration lors de son chargement lors d'une session. La fonction récupérait l'état du bouton du formulaire au lieu de sa valeur, le numéro du fichier. Ce qui rendait les logs incomplets.

BA1 - Bandeau de la page d'accueil

L'étape 3 de sélection du poids des attributs était manquante dans le bandeau de la page d'accueil, que j'ai ajouté à l'aide du logiciel d'édition d'images Gimp.

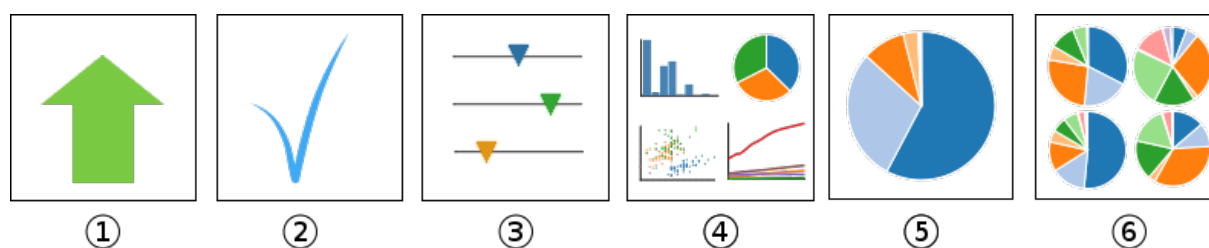


Figure 9 – BA1 - Bandeau de la page d'accueil après ajout de l'étape 3.

BA2 - Texte flottant sur les boutons des étapes

Simple correction syntaxique, quelques info-bulles ont été corrigées et correspondent désormais parfaitement aux titres des étapes du site.

Am1 - Textes d'aide

Plusieurs étapes du site nécessitaient une amélioration de l'aide intégrée à l'interface. Ayant refaite celle de l'étape de pré-sélection, un texte d'aide concis a été ajouté sous le titre et la description de la section, précédée d'une icône d'information. D'autres textes d'aide restent à rajouter aux autres étapes, le mieux étant de conserver le modèle pour plus de cohérence.

Ce texte est visible dans la figure 11 présentant la nouvelle interface utilisateur de l'étape 4.

Am2 - Sélection par défaut à l'étape 2

Cette amélioration avait pour objectif de faciliter la navigation en choisissant par défaut l'option activant tous les objectifs. Pour résoudre cela, il a fallu rajouter une propriété à l'élément HTML (checked="checked"), et tester cette valeur à l'initialisation du module RequireJS.

Résolue au début de la phase de développement, cette correction n'a pourtant pas été simple de par la structure même du projet, découpé en modules. En effet, lors d'un changement en étape 2 (choix de nouvelles données), le module est réinitialisé mais pas supprimé. Il a donc fallu comprendre ces méthodes et tout le système d'instanciation des modules. Ce problème est bel et bien résolu à ce jour, en ajoutant une nouvelle vérification de l'élément HTML lors de cette fonction de reset.

Am3 - Sélection multiple en étape 4

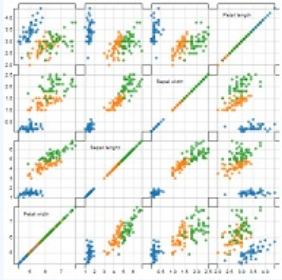
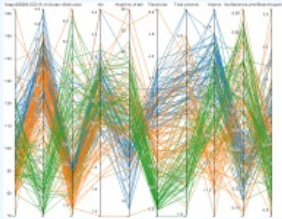
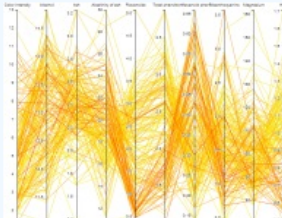
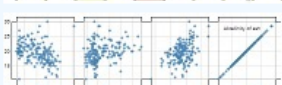
La sélection multiple de l'étape 4, permettant de pré-sélectionner les différentes visualisations désirées n'était pas très intuitive. Ainsi, les boutons "SELECT ALL" et "UNSELECT ALL" ont été remonté au dessus de la table et l'aspect de l'interface utilisateur a été revu pour mieux correspondre à la charte graphique du site. De plus, comme précisé dans le paragraphe correspondant, ça a été l'occasion d'ajouter le texte d'aide.

Les figures 10 et 11 permettent de rendre compte des changements apportés à l'interface.

Pre-select some visualizations

PREVIOUS
0
1
2
3
4
5
6
NEXT

Pre-select some visualizations that you favor or that best match your data and your objectives

Visualization thumbnail	Visualization name	Objectives score	Data to vis. mapping score
	4 scatter plots matrix with colors for class	30%	99%
	10 axes max parallel coordinates with class	29%	98%
	10 axes max parallel coordinates with measure	29%	76%
			

Select All
Unselect All
☐ Show "data to vis." mapping

Figure 10 – Am1 et Am3 - Ancienne UI de l'étape 4.

Pre-select some visualizations



Pre-select some visualizations that you favor or that best match your data and your objectives

 Click on rows to pre-select visualizations.

☐ Show "data to vis." mapping

Select All

Unselect All

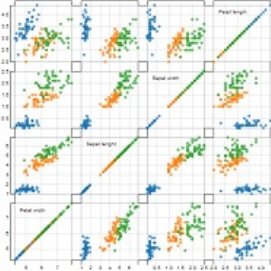
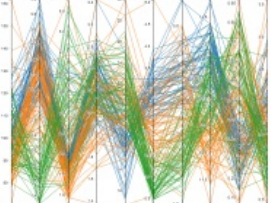
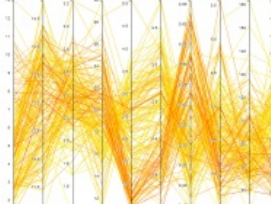
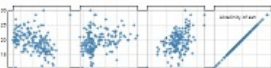
Visualization thumbnail	Visualization name	Objectives score	Data to vis. mapping score
	4 scatter plots matrix with colors for class	30%	98%
	10 axes max parallel coordinates with class	29%	98%
	10 axes max parallel coordinates with measure	29%	63%
			

Figure 11 – Am1 et Am3 - Nouvelle UI de l'étape 4 avec texte d'aide.

Am4 - Bouton EVALUATE

Le bouton EVALUATE présent en dernière étape à la place du bouton NEXT permettant d'aller à l'étape suivante dans le site provoquait une confusion. En effet, le premier était identique au second, en dehors du texte. Pour pallier ce problème, la couleur a été modifiée (Figure 12).

Interactive Genetic Algorithm



Figure 12 – Am4 - Bouton EVALUATE en étape 6 changé en vert.

9

Campagne de tests et débogage

1 Jeux de test

Afin de tester l'application tout au long du développement, des fichiers de démonstration y sont intégrés. Ils présentent chacun des caractéristiques différentes en terme de taille, de type et de nombre d'attributs. Voici les jeux de données que j'ai principalement utilisés et pourquoi :

- Wine et French Demography : Pour des tests de nombreux attributs numériques. Utilisé pour vérifier le temps de chargement et la lisibilité des histogrammes.
- Box Office Mojo : Plusieurs attributs nominaux (urls, images, ...)
- CIA World Fact Book : Combinaison de deux attributs, un numérique et l'autre nominal (noms de pays).
- Iris : Peu d'attributs à la représentation simple. Trois classes pour l'attribut nominal et 4 attributs numériques très différenciés selon la classe. Utilisé pour vérifier la bonne visualisation et rapide pour vérifier le bon fonctionnement de l'UI.

2 Méthode de test

A chaque ajout de fonctionnalité mineure ou majeure, un roulement de tous les jeux de tests sus-nommés dans l'application est effectué. On va alors vérifier en premier lieu si l'affichage est correct, puis, en faisant des aller retours dans les modules, tester leur intégrité.

1. Pour chaque jeu de test
 - (a) Parcours du site jusqu'au dernier module
 - (b) Affichage des différentes vues concernées par la modification
 - (c) Retour à l'étape 2 et changement de jeu de test

Plusieurs scénarios sont alors possibles :

- L'intégrité des modules est menacée, par exemple lorsqu'un aller-retour provoque un plantage du site
- Un module ne se charge pas correctement
- Les données visualisées ne sont pas conformes aux attentes
- Un affichage ne fonctionne pas comme prévu

Dans les deux premiers cas, le débogage se fait en direct grâce à la console JavaScript du navigateur. Les erreurs y apparaissent et c'est généralement une simple erreur de code. Ce fut le cas pour les variables locales qu'il fallait passer en attribut du module Variables. Si les données visualisées ne sont pas conformes aux attentes, alors on passe en mode débogage grâce à la console, en surveillant les données. Enfin, pour l'affichage, ce sont généralement des erreurs de code CSS / HTML, voire d3js en début de développement.

3 Rapports de bugs

Chaque semaine, un compte-rendu de l'avancée était réalisé auprès de M. Venturini, avant de mettre en ligne les modifications sur l'instance de test à l'adresse www.vizassist.com/PRDtest. En parallèle de ma procédure de test fonctionnels, mon encadrant faisait régulièrement ses propres tests le reste de la semaine et me fournissait un rapport de bugs supplémentaire au début de la séance suivante.

Voici quelques-uns de ces rapports :

```

1 Rapport de bugs du 15/03/16 :
2
3 Détails à changer :
4 - Etape 3, dans l'affichage "values", enlever occurrence (qui ne semble pas avoir de sens ?)
5 - Mettre a jour le numéro de version
6
7 Bugs :
8
9 - Etape 2 : J'ai un peu de mal a cliquer dans les cases a cocher c'est bizarre, parfois ↩
    ca marche, d'autre fois non
10 - Etape 6 : impossible de cliquer sur les individus ou de cocher leurs cases

```

```

1 Rapport de bugs du 29/02/16 :
2
3 - Plantage de l'application au changement de la base Pima vers Iris et raffichage des ↩
    attributs.
4 - Pour les attributs non numériques (nominaux, etc), la visu n est pas la bonne : les ↩
    barres devraient toutes avoir la meme hauteur (mettre une hauteur qui depend de ↩
    l indice n a pas trop de sens, en plus en changeant l ordre , ça fait ↩
    apparaitre des barres etc, voir tests fichier vizassistproject). Idealement : ↩
    barre de meme hauteur avec une couleur qui depend de la valeur.
5 - Attention, gros pb : quand je change les valeurs d importance , l affichage du ↩
    graphique change aussi

```

```

1 Rapport de bugs du 25/02/16 :
2
3 - Petit bug sur le bouton next non actif au moment de choisir les objectifs + cochage des ↩
    cases quand on revient en arriere
4 - Toujours un pb conceptuel avec values
5   - Elles sont triées par ordre croissant pour chaque attribut ? ce n est pas tres ↩
    interessant du coup (pas tres informatif)
6   - Il vaudrait mieux trier toute les données selon une colonne et afficher l ordre ↩
    qui en resulte ensuite Drop down list apres "values": Values sorted by
7   - Exemple : trier par la 1ere variable nominale ou ordinale ou time (en tenant compte ↩
    des importances ? mais bon la ca reboucle)
8   - Et sinon laisser tomber values
9   - Pb avec base Iris et l affichage des valeurs nominales de l attribut class => ↩
    j ai l impressin que c est lorsque l attribut classe prend des ↩
    valeurs autres q u entieres -> a priori a traiter comme string.
10 - Dans l affichage on mouse over : les [] devraient etre [ ? Egalement, on trouve ↩
    des valeurs uniques plut t que des intervalles, c est bizarre

```

10

Gestion de projet et reprise

1 Gestion du temps

La figure 1 décrit le gantt réel. Les cases grises au contour noir représentent les étapes du planning prévisionnel et permettent d'y confronter la réalité en fin de projet.

On constate donc que le second sprint dédié au développement du Dashboard n'a pas été réalisé. A la place, on a un léger décalage d'une semaine sur le départ du premier sprint pour la visualisation des attributs, et qui va se prolonger jusqu'à la fin. Ces dernières semaines ont beaucoup servi au débogage et aux diverses améliorations. De plus, les quelques correctifs qui ont été réalisés se sont retrouvés un peu éparpillés avant et après le premier sprint.

Enfin, la phase de rédaction du rapport et de préparation de la soutenance se retrouve avancée d'une semaine. Cela est dû au fait que nous n'avions pas toutes les précisions concernant ces dates.

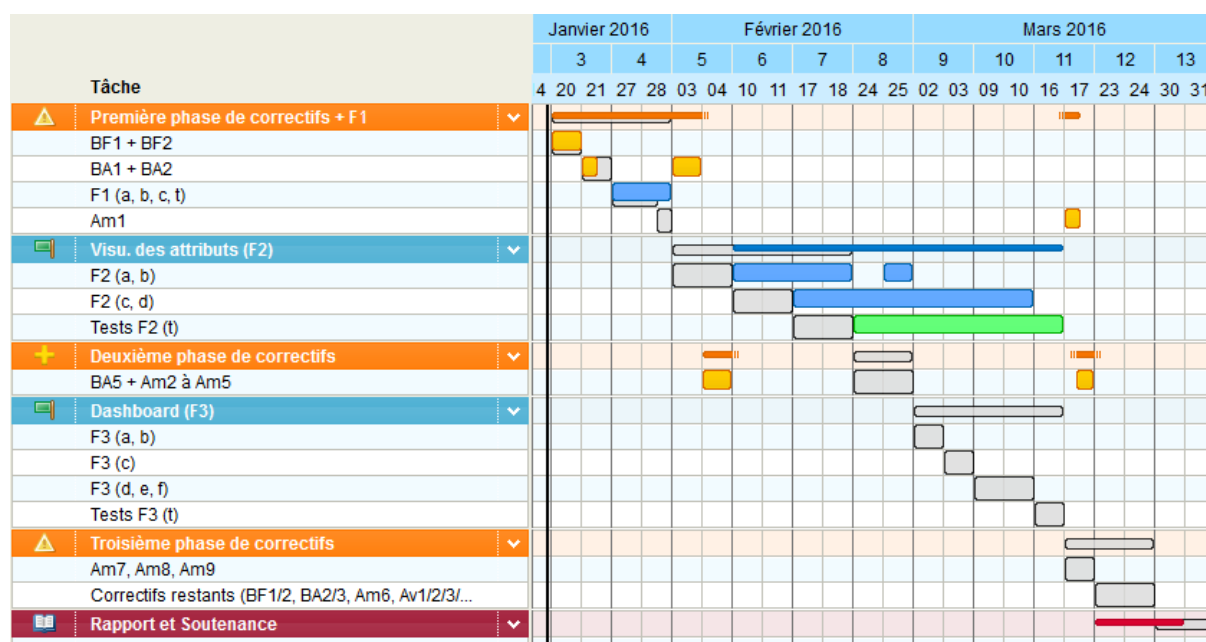


Figure 1 – Planning réel vis à vis du planning prévisionnel (tronçons noirs)

Le développement du Dashboard a été abandonné d'un commun accord avec M. Venturini vers le milieu du premier sprint, constatant le temps nécessaire pour mener à bien l'implémentation de la visualisation

des attributs. Proposant plus d'ajouts à cette fonctionnalités que prévues dans les spécifications au gré des retours de mon encadrant, nous avons préféré produire quelque chose de fiable et de complet.

2 Procédure de versionning

Lors des projets précédents liés à VizAssist, un dépôt git avait été créé et une procédure précise mise en place. Celle-ci n'était présente que dans le rapport de PFE de Yann-Thomas Le Moigne^[1], dans la section dédiée à la reprise. De plus, quelques modifications liées aux changements de versions du site sont à faire dans deux fichiers du projet.

Afin d'améliorer la lisibilité de cette procédure, je l'ai rajouté dans le fichier README.md du projet, en complément des diverses informations qu'il contient (notamment des conventions de nomage, etc.).

Procédure de versioning telle que rajoutée dans la documentation du projet

```

1  ### Versionning
2
3  # Git
4  Clone the master's branch of the project
5  1. Create/join a dev branch : git checkout -b dev origin/dev
6  2. Make your changes
7      * Comment the googleanalytics line before any local test : index.php (line 34)
8  3. Duplicate your dev branch and update the version number :
9      * Create the branch : checkout -b release/v1.x.y dev
10     * Update the version references (see below) and commit :
11         git commit -a -m "Version 1.x.y"
12  4. Merge to the master
13     * Go to the master branch : git checkout master
14     * Merge : git merge no -ff release/v1.x.y
15     * Uncomment the googleanalytics line (see step (2.)) and commit
16  5. Update the dev branch
17     * Go to dev branch : git checkout dev
18     * Merge with the temporary branch : git merge no -ff release/v1.x.y
19     * Delete the temporary branch : git branch -d release/v1.x.y
20  6. Push your changes : git push
21
22  # References
23  When you update the project, incremente the version according to
24  your modifications.
25  1. Changelog (Below in this document)
26  2. js/main.js (version number - line 5)
27  3. js/main.js (date of release - line 80)

```

Cette procédure n'a pas été suivie correctement au tout départ du projet. En effet, du fait de son absence de la documentation interne au projet et des informations qui m'avaient été transmises, j'avais créé une branche ne respectant pas ces conventions. J'ai toutefois rapidement corrigé cela aussitôt la procédure intégrée.

A terme, j'ai travaillé sur la branche "dev" prévue, laquelle est copiée avant la mise en production vers une branche "release/1.6.0" qui sera fusionnée avec la branche principale. Cette dernière étant, au moment du push, synchronisée avec le serveur de la version en ligne.

Changelog

Afin de suivre les modifications effectuées au fur et à mesure des versions, trois éléments sont donc à rajouter dans la documentation interne : le numéro de version, la date de mise en production et le changelog. Ce dernier étant également inclut dans le fichier README.md :

```

1  _Version 1.6.0:_
2  * Added a visualisation of attributes in the weight table (distribution and values).
3  * Fixed tooltips texts in the breadcrump.
4  * "I don't know" is now checked by default on the objectives module
5  * Changed the aspect of the EVALUATE button at step 6
6  * Improved the layout of the pre-selection step.
7
8  _Version 1.5.4:_
9  * Added a Donut Chart Vizualisation (Nominal & Ordinal)
10 * Fix issue with logs when a demo file is selected
11 * Completed README.md with versioning
12
13 _Version 1.5.3:_
14 * Corrections in PHP (static methods)
15 * Fix issue in log writing when selecting a demo file
16
17 _Version 1.5.2:_
18 * Added a description in the variables editor
19 * Welcome page updated with the "Adjust attributes" step

```

3 Correctifs restants

Afin d'aiguiller une possible reprise du projet VizAssist, cette section fait état des bugs et problèmes identifiés à ce jour et n'ayant pas été corrigés. Les différents tests effectués sur les ajouts, notamment de la visualisation des attributs, ont permis de révéler plusieurs problèmes qui ont été corrigés depuis. On ne trouvera donc ici que la liste des éléments identifiés lors de la phase de recherche et d'analyse et il faut garder en tête que celle-ci ne sera jamais exhaustive et il est bien évidemment possible d'en identifier de nouveaux.

Ces listes utilisant les dénominations propres à ce rapport, référez-vous à la section 2 (Chapitre 6) des spécifications pour davantage de détails.

Correctifs partiellement appliqués

- **Am1** - Textes d'aide
- **Am5** - Titre des sections

Problèmes non résolus

- **BF3** - Suggestions des visualisations non conformes
- **BF4** - Time Series Interactive Graph
- **BA3** - Texte coupé sur certains axes de certaines visualisations
- **BA4** - Cluster Dendogram : Croisements non désirés
- **BA5** - Box Plot for nominal attribute : Labels
- **Am6** - Affichage de la légende
- **Am7** - Nom des visualisation en étape 5
- **Am8** - Nom du jeu de données
- **Am9** - Dispositions en étape 1
- **Av1** - Bubble Tree et Zoomable Treemap
- **Av2** - (Un)Directed Graph force-directed layout
- **Av3** - Multiline chart with brushing
- **Av4** - Small multiples time series



Conclusion

Pour la partie recherche du projet, compte tenu de la nature de celui-ci, les travaux se sont principalement orientés sur l'aspect conception et élaboration d'interfaces utilisateurs. Sonja Rohmer ayant fait une étude très complète dans son document, j'ai simplement fait une synthèse de celle-ci avec VizAssist au centre. J'ai également tenté de faire une mise à jour, mais ayant été rédigé moins d'un an plus tôt, presque rien n'a encore changé.

J'ai donc cherché à apporter autre chose à la partie recherche en réalisant de ma propre initiative des tests d'utilisabilité en me basant sur des cours suivis lors de mon semestre à l'étranger. Je pense que cela a été très utile pour mettre en lumière certains dysfonctionnements peu visibles de l'extérieur et souvent difficile à faire exprimer par les utilisateurs. De plus, beaucoup de maquettage a été fait et la recherche de solution avec force de propositions m'a semblé s'insérer dans les enjeux de cet exercice.

C'est donc avec un cahier de spécification très rempli et un planning très serré qu'a démarré la partie développement. Les questions que l'ont pouvait se poser concernait le temps nécessaire pour la réalisation de tous ces objectifs et malgré un planning prévisionnel très détaillé, du retard s'est fait ressentir dès les premières semaines. Finalement, seules les deux premières fonctionnalités, et notamment la visualisation des attributs, ont été implémentées, laissant volontairement de côté le Dashboard. En effet, lorsque la moitié du temps fut écoulé, il a été décidé d'un commun accord de se concentrer sur le débogage et la complétion de ce qui avait été fait, afin de pouvoir mettre en production les travaux de ce projet.

Très satisfait du déroulé du projet dans son ensemble, mes seuls regrets se tournent vers la fonctionnalité non développée ainsi que les nombreux bugs et problèmes soulevés qui n'ont pu être résolus. Je pense toutefois avoir produit des ajouts pertinents et fonctionnels à la plateforme VizAssist au regard du temps qui m'a été accordé. Bien sûr, toute la spécification faite en amont pourra être utilisée pour de futurs collaborateurs et tous ces travaux de recherche n'auront pas été vains !



Comptes rendus hebdomadaires

Compte rendu n°1 du 17/09/2015

Cette première semaine fut consacrée à la découverte du sujet et de la plate-forme VizAssist. De nombreuses expérimentations de l'outil ont été faites pour bien en saisir le fonctionnement. Recherche d'informations sur les technologies : Documentation sur la bibliothèque D3js et lecture d'un premier tutoriel. Parcours du site www.d3js.org et des visualisations au code source accessible.

Compte rendu n°2 du 24/09/2015

Récupération du code source du projet depuis un gestionnaire de FTP, puis du rapport de PFE précédent contenant les instructions de reprises : accès administrateur et git. Récupération du dépôt proprement. Très légères modifications sur la branche principale (corrections orthographiques) et mise à jour du serveur en production avec ces corrections mineures. Lecture du document de Sonja Rohmer.

Compte rendu n°3 du 01/10/2015

Afin de faire fonctionner le site en local, un problème de compatibilité au niveau de PHP s'est posé. Un certain temps a été consacré à l'identification de cette erreur. En parallèle, parcours des visualisations existantes qu'il serait possible d'ajouter. L'erreur PHP venant de fonctions déclarées statiques dans une version non à jour de PHP, il a fallu corriger l'ensemble du code ayant cette erreur. La version en production a été mise à jour avec cette correction mineure. Enfin, création d'une branche PRDAntoine sur le git et création d'une instance de test en parallèle de la version de production sur le serveur à l'adresse www.vizassist.fr/PRDtest.

Compte rendu n°4 du 08/10/2015

Recherches en ligne sur les outils similaires à VizAssist en prévision de la réalisation de l'état de l'art. Comparaison de plusieurs sites contenus dans l'étude réalisée par Sonja Rohmer dans son document et mise à jour de certaines informations.

Compte rendu n°5 du 15/10/2015

Revue de code du projet. Utilisant la librairie RequireJS, il a fallu comprendre son fonctionnement et identifier les différentes fonctions du programme. De nombreux tests ont été effectués en local et sur l'instance PRDtest.

Compte rendu n°6 du 22/10/2015

Un bug majeur est apparu sur le serveur de production cette semaine ci. Les logs enregistrent les actions des utilisateurs ne contenant plus toutes les informations nécessaires, et ce quelques jours après une nouvelle diffusion de la plate-forme par M. Venturini. J'ai donc passé un certain temps à identifier le problème puis à le corriger. Le reste du temps a été consacré à la suite de la revue du code initiée la semaine précédente.

Compte rendu n°7 du 05/11/2015

Réunion avec M. Venturini pour une mise au point sur les objectifs. Nous avons ainsi défini la fonctionnalité de visualisation des attributs. A cette suite, la première maquette a été faite au crayon. Puis, recherche d'un type de visualisation à ajouter à la bibliothèque au regard des objectifs. La visualisation "Donut Chart" est choisie puis appréhendée.

Compte rendu n°8 du 12/11/2015

Spécifications complètes de l'ajout de la visualisation "Donut Chart" : travail de compréhension sur le code source du projet pour la gestion de la bibliothèque des visualisations avec les 3 fichiers liés à cette fonction, puis récupération du script D3js et des informations nécessaires à cet ajout.

Compte rendu n°9 du 19/11/2015

Récupération de la liste des bugs relevés par M. Venturini, complétée par ceux que j'ai trouvés moi-même. Un tri a été fait ainsi qu'un classement par catégorie après avoir reproduit chacun d'eux. Un maquetage plus complet est réalisé pour la visualisation des attributs conformément au brouillon, ainsi qu'une recherche sur le code D3js à intégrer pour cette fonctionnalité. Une partie de la seconde journée a été consacrée à la rédaction du rapport (reprise du modèle LaTeX, élaboration d'un plan, ...).

Compte rendu n°10 du 26/11/2015

Mise au point avec M. Venturini sur ses attentes pour la gestion des sauvegardes. Recherche ergonomique et création de 5 maquettes différentes à cet effet pour cette fonctionnalité. La seconde journée était consacrée au forum des entreprises.

Compte rendu n°11 du 03/12/2015

Préparation puis réalisation de tests d'utilisabilité auprès de 4 étudiants de Polytech au cours de la journée. Mesures, notes d'observations et analyse du comportement pour chacun d'eux effectué. Les résultats des tests ont été étudiés le lendemain. Deux créneaux ont été pris pour des entretiens auprès d'entreprises pour la recherche de stage (Worldline - mercredi 17h, Cap Gemini - jeudi 10h).

Compte rendu n°12 du 10/12/2015

Une liste de corrections s'est ajoutée à la liste existante suite aux résultats des tests d'utilisabilité. Une nouvelle maquette pour le dashboard a été réalisée conformément aux remarques de M. Venturini. La rédaction du rapport et notamment des spécifications s'est poursuivie.

Compte rendu n°13 du 17/12/2015

Trois nouvelles maquettes ont été réalisées au cours de la semaine au fur et à mesure des remarques M. Venturini. La dernière semble convenir et fait un bon compromis avec ses exigences et les fonctionnalités que le dashboard doit satisfaire. Rédaction du rapport (plan plus détaillé, poster et partie conception).

Compte rendu n°14 du 21/01/2016

Comme prévu sur le planning, les premières corrections essentielles ont été réalisées lors des premiers jours de développement. A savoir : BF1 (correction des erreurs PHP dues à des fonctions static), BF2 (Problème de log) et BA1 (image manquante dans le bandeau de la page d'accueil). Toutes ces corrections relativement mineures ont été testées rapidement avant d'être mises en production.

Compte rendu n°15 du 28/01/2016

Développement de la fonctionnalité F1 - ajout d'une visualisation Donut Chart. Etant donné toute la spécification préalable, cela fut rapide à rajouter et après quelques tests sur l'instance PRDTest du site, cette visualisation a été validée.

Compte rendu n°16 du 04/02/2016

Complétion du fichier README pour le versionning afin d'intégrer la procédure décrite dans le rapport de Yann-Thomas Le Moigne. Correction des branches du git afin de s'y soumettre et mise en production de la version 1.5.4. Deux nouvelles corrections : BA2 (Mise en correspondance des textes des étapes dans les infobulles) et AM2 (Sélection par défaut de l'option "I don't know"). Cette dernière a prit plus de temps que prévu dû à la spécificité de l'architecture voulue par RequireJS.

Compte rendu n°17 du 11/02/2016

Début du sprint de développement de la fonctionnalité F2, la visualisation des attributs. Création pour cela d'une page de test pour appréhender le script d3js d'histogramme trouvé lors des spécifications. Au vu de la complexité inutile de ce dernier, développement intégral d'un nouveau script grâce auquel les valeurs des données sont affichées pour les attributs numériques uniquement.

Compte rendu n°18 du 25/02/2016

Après la mise en ligne d'une instance de test du script de visualisation des attributs, un premier rapport de bug m'a été transmit. Ajout de la visualisation de la distribution à la place des données et amélioration générale de l'affichage de la table des attributs en regroupant certains éléments. Quelques corrections fonctionnelles suite aux retours. Il est convenu que le développement s'arrêtera à cette fonctionnalité, le "Dashboard" nécessitant beaucoup trop de temps.

Compte rendu n°19 du 03/03/2016

Après discussions, il s'avère qu'une visualisation des valeurs est pertinente en complément de la répartition à condition de pouvoir les trier. Ajout d'un tri sur les valeurs avec une vue séparée distribution/valeurs. Ajout d'une info-bulle sur les barres des histogrammes indiquant les valeurs ou les intervalles de valeurs (pour la distribution). Ajout des couleurs pour les valeurs non numérique plutôt qu'une valeur de "hauteur" arbitraire.

Compte rendu n°20 du 10/03/2016

Suite aux nouveaux rapports de bug, beaucoup de corrections et d'ajustements. Les intervalles de valeurs sont plus fidèles aux valeurs pour la distribution, le haut des barres de l'histogramme est plein afin d'avoir un meilleur accès aux info-bulles. Les variables liées à l'étape sont maintenant correctement réinitialisées lors d'un changement de données. Ce dernier problème était connu depuis plusieurs jours et a été difficilement identifié. Il venait d'une mauvaise instanciation du module RequireJS, les variables ont été passées en attributs et les méthodes de réinitialisation ajustées.

Compte rendu n°21 du 17/03/2016

Nouveau rapport de bugs. Correction d'un problème de clic bloqué sur les checkbox en étapes 2 et 6. Correction de la sélection automatique en étape 2 qui était aussi lié à l'instanciation des modules RequireJS. Corrections orthographiques et refactorisation d'une partie du code. Nouvelle série de correctifs : AM1 (textes d'aide ajoutés), AM3 (déplacement des boutons de sélection de l'étape 4) et AM4 (modification du bouton EVALUATE en étape 6 pour le différencier des boutons NEXT et PREVIOUS). Révision globale de l'UI de l'étape 4 suite aux changements apportés par les corrections AM1 et AM3.

Compte rendu n°22 du 24/03/2016

Nettoyage du code, rédaction du rapport et mise en production de la version 1.6.0.

Compte rendu n°23 du 30/03/2016

Soutenance finale.



Webographie

- [WWW1] FUSIONCHARTS. *Site de téléchargement d'un outil payant similaire à VizAssist proposant une comparaison des différentes bibliothèques graphiques JavaScript.* URL : <http://www.fusioncharts.com/javascript-charting-comparison/> (visité le 19/11/2015).



Bibliographie

- [1] Yann-Thomas Le MOIGNE. « Rapport de PFE : Assistant web pour la visualisation de données - VizAssist ». Mém.de mast. Polytech Tours, mai 2015.
- [2] Sonja ROHMER. « Web interfaces and assistants that support data visualization ». Mém.de mast. Polytech Tours, jan. 2015.

Amélioration de la plate-forme d'aide à la visualisation de données VizAssist

Antoine Fonfria

Encadrement : Gilles Venturini Objectifs

VizAssist est un outil en ligne d'assistance à la visualisation de données. Les objectifs de ce PRD sont :

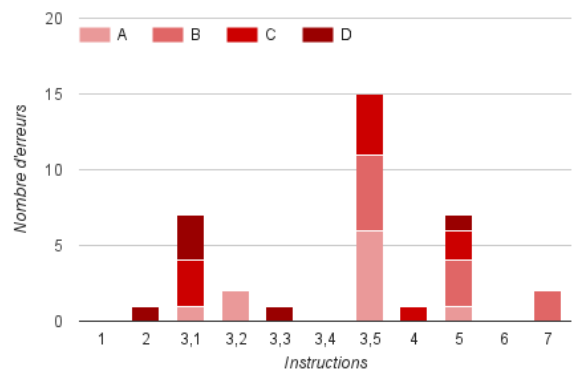
- Identification des problèmes (bugs, ergonomie de l'UI, ...).
- Maintenance et corrections.
- Ajout d'une visualisation d'attributs.
- Ajout d'un tableau de bord.

Analyse et corrections

En parallèle de la correction de bugs, retouche de l'interface utilisateur sur de nombreux points, avec notamment l'ajout de textes d'aide, le déplacement de certaines composantes, la complétion de menus, etc. Mise en place de tests d'utilisabilité permettant de repérer les dysfonctionnement de l'interface existante en y confrontant de nouveaux utilisateurs.

Ajouts fonctionnels

Création d'un système de tableau de bord, permettant comparaison et sauvegarde de visualisations et ajout d'une représentation de la répartition des attributs dans les jeux de données. Réalisation progressive de maquettes. Intégration d'un type de visualisation supplémentaire à l'outil. Utilisation de la bibliothèque graphique D3.js.



Extrait des résultats des tests d'utilisabilité



Maquette du tableau de bord à implémenter

Amélioration de la plate-forme d'aide à la visualisation de données VizAssist

Résumé

Ce Projet Recherche et Développement a pour objectif d'apporter des améliorations à la plate-forme en ligne gratuite d'aide à la visualisation de données VizAssist, créée au travers de multiples travaux au sein de l'école, encadrés par M. Gilles Venturini. Cette amélioration passera ainsi par une maintenance de l'existant, des corrections et l'ajout de fonctionnalités : nouveaux choix de graphiques, une visualisation de la répartition d'attributs dans un jeu de données et un système de tableau de bord pour sauvegarder des configurations. L'application étant sous la forme d'un site internet, elle utilise les technologies web classiques : HTML, CSS, JavaScript et PHP. Elle met également à profit les bibliothèques JavaScript jQuery, RequireJS, et D3js pour les graphiques.

Mots-clés

VizAssist, Visualisation, Données, d3js, RequireJS

Abstract

This Research and Development Project has for objectives to make improvements to the free web helper for data visualisation VizAssist, a platform created through multiples projects in Polytech over the years and supervised by Mr Gilles Venturini. This improvement will consist in a maintenance of the existing website, some corrections and new features : a new choice of graphic visualization, a visualization of the attributes distribution in a dataset and a dashboard system for saving configurations. The application is a website, therefore it uses conventional web technologies : HTML, CSS, JavaScript and PHP. It also includes some JS libraries : jQuery, RequireJS and D3js for the graphics.

Keywords

VizAssist, Visualization, dataset, d3js, RequireJS

Tuteurs académiques

Gilles VENTURINI

Étudiants

Antoine FONFRIA (DI5)