

Eric Lecoutre

Outils informatiques pour le statisticien

Lecoutre, Eric:

Outils informatiques pour le statisticien

Institut de statistique

Université catholique de Louvain

Tous droits réservés.

© 2005 Eric Lecoutre, Louvain-la-Neuve

Date d'impression: 10 février 2005

Table des matières

I	Edition de documents, sorties logiciels et reporting	1
1	Quelques formats et outils pour l'édition	5
1.1	Le format texte	5
1.1.1	Tout un alphabet en 0 et 1	5
1.1.2	La table des caractères ASCII	5
1.1.3	Unicode	6
1.1.4	Editeur de texte	7
1.1.5	Tout débute avec le texte	7
1.2	Quand on imprime...	8
1.2.1	Le Postscript	8
1.2.2	Du Postscript au PDF	8
1.3	Le texte s'enrichit : RTF	10
1.4	Incursion L ^A T _E X	10
1.4.1	De T _E X à L ^A T _E X	10
1.4.2	LyX	11
1.5	Publication Internet - les pages web	11
1.5.1	Les débuts du web	11
1.5.2	Mises en forme avancées	12
1.5.3	Les feuilles de style	13
1.6	Les langages balisés	13
1.6.1	Retour aux sources : le SGML	13
1.6.2	XML	14
1.6.3	Quelques DTD	16
1.6.4	Quelques démonstrations supplémentaires	18
2	Quelques (bonnes) pratiques	19
2.1	De la sémantisation	19
2.1.1	Séparation forme/contenu	19
2.1.2	Structure sémantique	19
2.1.3	Métadonnées	19
2.2	Liens entre les formats	19
2.3	Literate Programming	20
2.4	Et Word dans tout ça ?	21
2.5	DOM et API	21
3	Pour les statisticiens	23
3.1	Reporting direct	23
3.1.1	SAS	23
3.1.2	SPSS	24
3.1.3	Splus SPXML	24
3.1.4	R2HTML	24
3.1.5	Projet ROMA	24
3.2	Rapports dynamiques	25

II Table des matières

3.2.1	Sweave	25
3.2.2	Rpad	25
3.2.3	SAS Office	25
3.3	Pour les données ?	25
3.3.1	Structuration des données	26


Première partie

Edition de documents, sorties logiciels et reporting

Introduction

La chaîne de la statistique appliquée comprend plusieurs étapes : l'acquisition des données (enquêtes, plans d'expérience, inventaires, ventes,...), la gestion des données (*data management*), l'analyse statistique à proprement parler, l'interprétation des sorties et finalement la communication des résultats. C'est de cette dernière activité dont il sera question ici.

Le présent support accompagne un exposé. Seul et sans ses exemples, il est donc incomplet et incompréhensible. La dernière version du document et une archive zip contenant les démonstrations se trouvent à l'adresse suivante :

 Outils Informatiques pour le statisticien - support et exemples

– <http://www.stat.ucl.ac.be/ISpersonnel/lecoutre/stats/>

L'organisation finale du document est prévue en trois grandes parties :

1. Edition de documents
2. Manipulation de données
3. Graphiques

Chaque partie dresse un aperçu des formats, outils et méthodes utiles au statisticien.

TAB. 1: Liste des sigles et acronymes utilisés

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CSS	Cascading Style Sheet
CSV	Comma Separated Value
DTD	Document Type Definition
DVI	Device Independant
EPS	Encapsulated Postscript
HTML	HyperText Markup Language
IDE	Integrated Development Environment
ODS	Output Delivery Sytem (SAS)
OMS	Output Management System (SPSS)
PDF	Portable Document Format
PS	Postscript
RTF	Rich Text Format
SGML	Standard Generalized Markup Language
SVG	Scalable Vector Graphics
UTF	Unicode Transformation Format
WYSIWYG	What You See Is What You Get
WYSIWYM	What You See Is What You Mean
XHTML	eXtensible HyperText Markup Language
XSLT	eXtended Stylesheet Language Transformations
XML	eXtensible Markup Language
XSL-FO	eXtended Stylesheet Language - Formating Object

Dans l'énumération précédente, il peut sembler surprenant de voir apparaître l'édition de documents en premier. Le chercheur (mémorant, doctorant, ...) pensera que la rédaction vient en dernier. L'activité de *reporting* aussi vient en dernier. Toutefois, je suis convaincu que la rédaction devrait toujours venir en premier, ne serait-ce qu'une description de la tâche à venir. Cette première partie vise à montrer qu'aucune rédaction ne devrait être prise à la légère.

Au préalable à toute rédaction, voici une liste de questions que l'on peut se poser :

Quel contenu ? Principalement du texte, des graphiques et des tableaux. En principe : des phrases, une introduction, une conclusion

Pour qui ? Note personnelle, interne à un département, destinée à l'Intranet ou bien publique, officielle...

Quel média ? Rapport destiné à une lecture sur écran, à une impression ou encore à une projection.

Une charte ? Forme libre ou respect d'une charte d'entreprise. Il faut garder à l'esprit que la charte peut changer un jour...

Pérennité ? Lecture unique, documentation de référence, document collaboratif devant être modifié, livre,...

En principe, la réponse à ces questions va suggérer l'outil ou le format à utiliser pour la rédaction des résultats.

Mais : Connait-on bien les différentes possibilités ? Dispose-t-on des logiciels adéquats ? Dispose-t-on des compétences adéquates ?

En pratique : La majorité des documents électroniques sont mis en forme avec Word (ou produit équivalent). Et cela pose beaucoup de problèmes...

Chapitre 1

Quelques formats et outils pour l'édition

1.1 Le format texte

1.1.1 Tout un alphabet en 0 et 1

Petit rappel : un ordinateur travaille en binaire, il ne manipule que des 0 et des 1 (*bits*). Tout document sauvegardé sur disque dur, disquette ou CD sera "physiquement" une suite de 0 et 1 (des "creux" et des "bosses"). Puisque l'on ne dispose que de ces deux lettres fondamentales, il faudra en combiner plusieurs pour représenter un alphabet. Ainsi, la table 1.1 indique le nombre de caractères que l'on sait coder en utilisant n bits.

Mais au fait : de combien de caractères différents a-t-on besoin ? Au moins : 26 lettres de l'alphabet standard, fois 2 (minuscules et majuscules), les 10 chiffres, la ponctuation, les lettres accentuées, les caractères imprimables spéciaux (*,£,%,...), les caractères non imprimables (espace, tabulation, retour chariot), etc. On approche très vite la centaine de caractères. Puisque cela nécessite donc au moins 7 bits, autant en utiliser 8, qui font un octet et permettront de rajouter certains caractères spéciaux tels que "Æ", "ñ" ou encore "¿".

1.1.2 La table des caractères ASCII

Une norme a été mise en place pour fixer une bonne fois pour toute une table reprenant l'ensemble des caractères usuels, afin de permettre la communication entre différents ordinateurs / systèmes d'exploitation : il s'agit de la table ASCII (*American Standard Code for Information Interchange*). Dans la table ASCII

TAB. 1.1: Nombre de caractères représentables avec n bits

<i>bits</i>	Taille de l'alphabet	
1	2^1	2
2	2^2	4
3	2^3	8
4	2^4	16
5	2^5	32
6	2^6	64
7	2^7	128
8	2^8	256

TAB. 1.2: Quelques caractères de la table ASCII

Code	Hexa.	Caractère
0	00	NUL (null)
2	02	STX (début de texte)
9	09	TAB(ulation)
27	1B	ESC(ape)
32	20	Espace
48	30	0
65	41	A
97	61	a
126	7E	~

chaque caractère se voit ainsi attribuer un numéro unique¹. A titre informatif, la table 1.2 fournit les codes de certains caractères².

Les fichiers ne contenant que des caractères ASCII sont des fichiers texte. Traditionnellement, un fichier texte ASCII porte l'extension TXT.

Connaître certains caractères peut être utiles. Sous Windows, la majorité des programmes acceptent la saisie d'un caractère quelconque de la table au moyen de la combinaison : ALT+numéro, où *numéro* est spécifié en décimal. Formellement parlant, la table ASCII comprend uniquement les 128 premiers caractères (soit ce qu'on peut coder avec 7 bits). Le passage à 8 bits, qui apporte 128 nouveaux caractères est l'encodage latin-1.

☞ Table ASCII

– <http://www.lookuptables.com/>

1.1.3 Unicode

Unicode est le successeur encore en développement de la table ASCII. Ce standard a le but ambitieux d'attribuer un identifiant numérique unique à tout caractère ou glyphe utilisé dans le monde, indépendamment d'une plate-forme ou d'un logiciel. La première publication parlant de l'Unicode remonte à 1991. La version actuelle de la norme est la 4.0.1, datée de mars 2004. La version 3.2 recensait 95221 caractères, symboles et directives. Représenter autant de symboles nécessite plus des 7 bits requis par l'ASCII. Actuellement, on prévoit jusqu'à l'utilisation de 20 ou 21 bits par caractères. Pour des raisons d'efficacité (ne pas encombrer inutilement la mémoire), l'Unicode se décline en trois versions :

UTF-8 Les caractères sont codés sur 8 bits afin d'assurer une compatibilité avec les anciens standards (ASCII et latin-1). Les caractères moins utilisés seront codés via deux symboles, ce qui implique que les manipulations sur le texte (recherches, remplacements) demanderont plus de temps. Ainsi, en UTF-8, le symbole "é" est codé par "Ã©". Cependant, la bonne couverture des caractères de base et sa compatibilité avec les anciens standards contribuent à son succès, principalement pour l'édition web.

UTF-16 Les caractères sont codés sur 16 bits. La très grande majorité des caractères Unicode assignés pour l'instant, qui sont les caractères les plus fréquemment utilisés, peut être représentée sur 16 bits. Le langage Java utilise cette norme en interne.

UTF-32 Les caractères sont codés sur 32 bits, ce qui requiert beaucoup de mémoire (un simple "l" nécessitera 32 bits). Seul cet encodage sait toutefois gérer efficacement des documents mixant chinois, hébreux,...

☞ Quelques tables unicode [en]

– <http://www.alanwood.net/unicode/>

¹ En fait, on distinguera la table ASCII de base utilisant 7 bits et la table ASCII étendue ajoutant 128 caractères grâce à l'ajout du 8ème bit, dont il existe plusieurs variantes.

² Qui montrent pourquoi dans SAS on utilisera DLM='09'x pour spécifier la tabulation comme délimiteur

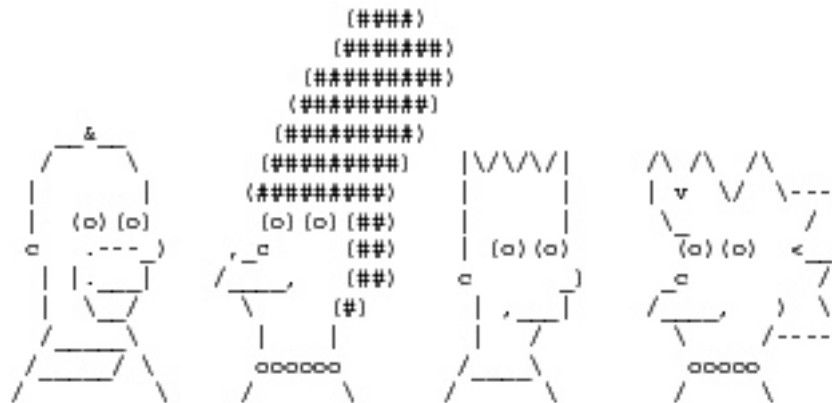


FIG. 1.1: ASCII art - démonstration

1.1.4 Editeur de texte

Attention à ne pas confondre éditeur de texte et traitement de texte. L'éditeur de texte se spécialise dans l'édition de documents au format texte. Le traitement de texte ajoute la mise en forme (Word est un traitement de texte). Pour le statisticien, l'éditeur de texte est très pratique car il permet de visualiser et modifier les jeux de données au format texte (fichier CSV par exemple). La plupart des éditeurs de texte sont en plus dotés d'outils supplémentaires transformant le logiciel en véritable couteau suisse : remplacements sélectifs (utilisation de *regular expressions*), sélections par blocs, enregistrement de macros pour automatiser le processus, etc. Certains éditeurs de texte sont orientés vers certaines applications : la programmation (C, C++, Fortran,...), l'édition de pages web, etc. Lorsque les outils spécifiques sont disponibles (compilation de code C à la volée par exemple), on parle alors plutôt d'IDE (*Integrated Development Interface*). L'institut utilise l'éditeur de texte Textpad (payant), mais il en existe d'autres, gratuits. On retiendra :

🔗 Textpad [en]

– <http://www.textpad.com/>
(shareware, très générique)

🔗 PSPad [en]

– <http://www.pspad.com/>
(freeware, plus orienté vers les pages web)

On trouvera une liste plus complète et une bonne définition à

🔗 Wikipedia : Text Editor [en]

– http://en.wikipedia.org/wiki/Text_editor

1.1.5 Tout débute avec le texte

Ascii Art

Au tout début, l'ordinateur ne disposait pas de capacités graphiques, mais se limitait à l'affichage de caractères (mode console, ligne de commandes). Pourtant, il était déjà possible de le doter de capacités ludiques. Jeux, mais aussi activités artistiques, comme en témoigne la mode de l'art ASCII. Le but est de faire de l'art, en se limitant à l'utilisation des caractères de la table ASCII (voir figure 1.1).

Les tables statistiques

En utilisant des "+" et des "-", il est assez rapide de mettre en forme des tables. Certaines tables ASCII étendues comprennent des caractères particuliers dédiés aux tables. C'est ce qui est utilisé par le système SAS

```

      TABLE OF YEARS BY HAPPY
    YEARS      HAPPY
Frequency|Low      |Medium  |High    | Total
-----+-----+-----+-----+
<=1yr   |      3 |      3 |      4 |  10
-----+-----+-----+-----+
1-3yr   |      2 |      2 |      8 |  12
-----+-----+-----+-----+
>3yr    |      1 |      0 |      2 |   3
-----+-----+-----+-----+
Total   |      6 |      5 |     14 |  25

```

FIG. 1.2: Exemple de sortie : une table en ASCII

pour fournir des tables plus agréables (bordures continues). Cependant, l'affichage correcte des sorties SAS dépend fortement de la présence des polices SAS. Beaucoup de logiciels fournissent encore des sorties texte.

Les graphiques en mode texte

Ils ont bel et bien existé... SAS ne produisait que cela pendant longtemps. La version 9 offre encore des possibilités de créer de tels graphiques, y compris en perspective isométrique (voir figure 1.3).

1.2 Quand on imprime...

L'impression d'un document depuis un logiciel quelconque fait appel au driver de l'imprimante pour gérer l'impression. Aujourd'hui, tous les drivers proposent l'option : impression dans un fichier. Regardons ce qui se passe de plus près avec un petit fichier texte.

1.2.1 Le Postscript

Les imprimantes de l'Institut sont toutes compatibles avec le langage PostScript. Il s'agit d'un langage de programmation développé par Adobe à partir de 1985 qui repose sur une formulation vectorielle des éléments. Sont inclus dans un seul fichier les éléments à imprimer, leur positionnement sur la page, les paramètres de la page (taille, marges), les couleurs etc. ; on parle actuellement de langage de description de page. Le langage en lui-même est indépendant du périphérique utilisé et le fichier pourra tout aussi bien être redirigé et traité par des imprimantes, mais aussi des traceurs ou des écrans³. Le côté vectoriel assurera la même qualité aussi bien sur une feuille A4 que sur un poster A0. Le langage postscript a principalement connu sa réussite sous les systèmes Unix (et Linux) dont beaucoup de programmes savent générer du postscript en natif (sans devoir passer par un driver d'imprimante spécifique).

1.2.2 Du Postscript au PDF

Tout le monde connaît aujourd'hui les fichiers PDF (Portable Document Format). Pourtant, au début, il ne s'agissait que d'un tour de passe-passe d'Adobe : PostScript et PDF sont en effet très proches. Le succès du PDF ? C'est d'abord une qualité d'impression doublée d'une qualité d'affichage. Normal, puisque le PDF

³ Cependant, la majorité des drivers des imprimantes actuelles ajoutent au fichier généré des codes de programmation propres à l'imprimante.

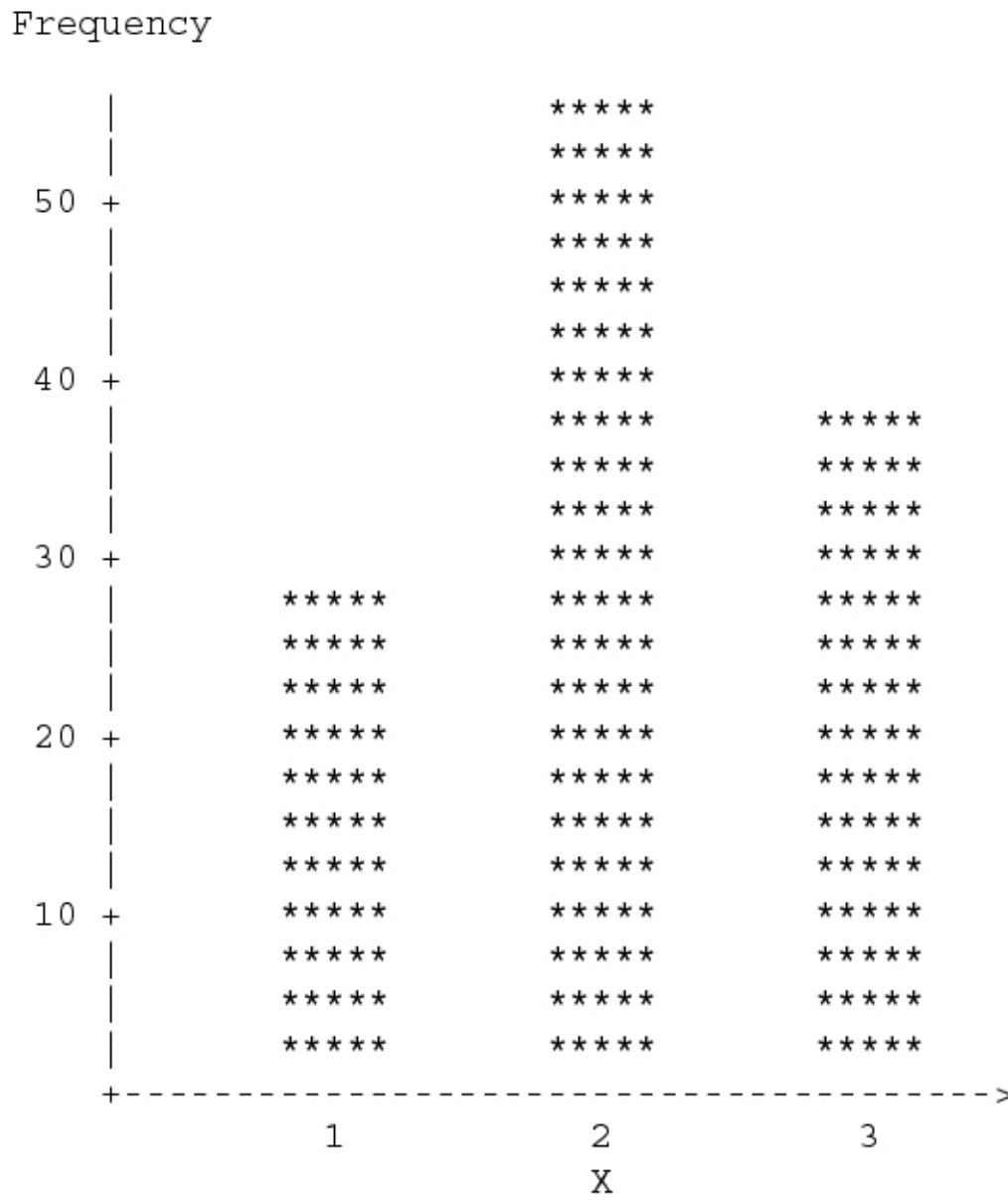


FIG. 1.3: Graph ASCII produit par la procédure SAS PROC CHART

garde le côté vectoriel du Postscript. C'est aussi une taille très légère (beaucoup de compression), la portabilité (toutes les plate-formes lisent le PDF) et des fonctions orientés web (inclusion facilitée d'images de divers formats, gestion des liens, table des matières, vignettes, etc). Enfin, si le format PDF a été créé par Adobe, il reste un format ouvert : chacun est libre d'en connaître les spécifications et peut écrire ses programmes qui vont générer à la volée des instructions PDF. On notera ainsi l'existence d'API gratuites pour les langages Java, Perl et PHP. Les Macintosh récents produisent du PDF en natif ; la suite bureautique Open Office offre la possibilité d'exporter tout document en PDF. Sur Windows, on pourra installer une imprimante virtuelle qui produira un PDF depuis toute application. Le principe est de produire un Postscript qui sera transformé en PDF.

Comparé au Postscript, le PDF a les avantages suivants :

- le langage se limite à un langage de description de page, là où le Postscript se posait en langage de programmation
- un fichier PDF n'est pas nécessairement linéaire : il y a pleins de pointeurs et l'information dont on a besoin peut être localisée n'importe où dans le fichier
- on peut mettre à jour un PDF en ajoutant du contenu à la fin
- certaines parties peuvent être compressées, voire cryptées
- il n'est pas nécessaire d'inclure les fontes de texte : le reader peut tenter de les émuler
- il vaut mieux l'éviter pour des raisons de portabilité, mais un fichier PDF peut inclure des commandes Postscript.

1.3 Le texte s'enrichit : RTF

L'application Wordpad permet d'améliorer un peu la présentation du texte en ajoutant des attributs de format : gras, italiques, alignements de paragraphes, etc. Wordpad exploite le format RTF, développé par Microsoft pour les premières versions de Word. Ce format est assez répandu ; ses spécifications sont en effet libres, bien que de plus en plus complexes au fil des versions. Il permet ainsi de servir de format pour échanger des fichiers Word entre plate-formes. Regardons ensemble le contenu d'un petit fichier RTF. Eh oui : le RTF, ce n'est que du texte, avec quelques mots clés servant de commandes, tel `\par` pour déclarer un paragraphe. Cela fait bien des années que Microsoft a développé un nouveau format pour Word, étant lui propriétaire et fermé ; le logiciel continue toutefois à proposer l'exportation (plus ou moins réussie...) en RTF.

1.4 Incursion \LaTeX

1.4.1 De \TeX à \LaTeX

En 1977, Donal Knuth, mathématicien et informaticien de génie, entame la rédaction du livre *The Art of Computer Programming*. Ne trouvant pas de système lui assurant une typographie de qualité suffisante, il décide mettre au point son propre système. Le langage de description formel \TeX naît alors, à la fois langage de description de page (comme Postscript) et logiciel qui interprète et "compile" ce langage. Le langage \TeX contient des commandes appelées primitives. L'utilisateur peut définir ses propres macro-commandes. Le succès de \TeX doit beaucoup à \LaTeX , un ensemble de macro-commandes développées par Leslie Lamport qui facilitent grandement l'utilisation de \TeX .

L'utilisation de \TeX (ou \LaTeX) passe par les étapes suivantes : écrire un document source (extension `tex`) contenant les macro-commandes de contenu/format et compiler ce document source. Cette étape peut faire intervenir différents compilateurs. On retiendra \LaTeX qui créera un fichier DVI (*Device INdependent*), lequel devra à son tour être converti en Postscript (via `dvi2ps`) et `pdf \LaTeX` qui créera directement un fichier PDF.

De nombreux logiciels facilitent l'écriture de documents \LaTeX . L'Institut utilise TeXnicCenter, véritable IDE s'intégrant immédiatement avec la distribution MiK \TeX .

 TeXnicCenter Home Page

– http://www.toolscenter.org/front_content.php?idcat=26

☞ \LaTeX à portée de clic

– <http://latex.perseguers.ch>

1.4.2 LyX

LyX est à la fois un format et un logiciel. Le format utilise des balises simplifiées et ne propose quasiment que des instructions de structure. Le logiciel permet un encodage aisé avec un aperçu visuel de la structure : ce n'est pas du WYSIWIG, mais du WYSIWIM (What You See Is What You Mean). LyX vient avec un convertisseur vers \LaTeX , qu'il utilise comme *back-end* pour l'impression et l'affichage à l'écran. LyX apparaît un produit idéal pour les débutants en \LaTeX : il facilite les utilisations avancées. Les images sont automatiquement converties en Postscript si nécessaire. Les équations sont affichées à la volée et un module graphique propose un accès à l'ensemble des opérateurs mathématiques.

☞ LyX - the Document Processor

– <http://www.lyx.org/>

1.5 Publication Internet - les pages web

1.5.1 Les débuts du web

Bref historique

1989-1992 Tim Berners-Lee, chercheur au CERN, annonce publiquement en août 1991 sur Usenet l'existence de la première page HTML (HyperText Markup Language), page autoréférente qui décrit ce qu'est le HTML. Nous reviendrons plus tard sur les origines du HTML. Pour l'instant, retenons que Tim Berners-Lee prévoit qu'une page HTML soit formée de différentes parties : le titre du document, des hyperliens, une structure sous forme de titres hiérarchisés, des listes et du texte brut.

1993 HTML est presque une norme (version 1), bien qu'évoluant encore rapidement. Petite révolution, le tag `img` fait son apparition.

1994 Un nouveau navigateur (Netscape) fait sensation : il inclut le support de fonctionnalités inédites tels que le centrage de texte, le clignotement. Les navigateurs s'empressent d'intégrer le support des tables, fort attendues des webmasters pour proposer des mises en page élaborées.

1997 Quelques années auront été perdues sur des développements arrêtés prématurément : il devient difficile de mettre d'accord les trois grands groupes suivants : développeurs de la norme, développeurs de navigateurs et webmasters. En janvier, la version 3.2 de la norme est terminée. Avec les quelques nouveaux éléments de présentation, cette norme va connaître beaucoup de succès et est encore très utilisée. A la fin de l'année, la version 4 ajoute le support des cadres (frames) et des objets (applets java par exemple).

1998-1999 Seules quelques corrections sont apportées, donnant lieu à la norme 4.01.

2000 à nos jours Les développements prennent une autre direction, dont on parlera plus tard.

HTML est un langage dit de "marquage" chargé de formaliser l'écriture d'un document avec des balises de formatage indiquant la façon dont doit être présenté le document et les liens qu'il établit avec d'autres documents. Il permet notamment la lecture de documents sur Internet à partir de machines différentes grâce au protocole HTTP, permettant d'accéder via le réseau à des documents repérés par une adresse unique, appelée URL. On appelle World Wide Web (noté WWW) ou tout simplement Web la "toile virtuelle" formée par les différents documents (appelés pour l'occasion pages web) liés entre-eux par des hyperliens.

Les pages web sont généralement organisées autour d'une page d'accueil, jouant un point central dans la navigation des visiteurs parmi les pages HTML à l'aide des liens hypertextes. Cet ensemble cohérent de pages web liées par des liens hypertextes et articulées autour d'une page d'accueil commune est appelée site web.

Le Web est ainsi une énorme archive vivante composée d'une myriade de sites web proposant des pages web pouvant contenir du texte mis en forme, des images, des sons, des vidéos, etc.

Il est important de comprendre que le langage HTML est un standard, c'est-à-dire qu'il s'agit de recommandations publiées par un consortium international, : le World Wide Web Consortium (W3C). Les spécifications officielles du HTML décrivent donc les "instructions" HTML mais en aucun cas leur implémentation, c'est-à-dire leur traduction en programmes d'ordinateur, afin de permettre la consultation de pages web indépendamment du système d'exploitation ou de l'architecture de l'ordinateur. Toutefois, aussi étoffées les spécifications soient-elles, il existe toujours une marge d'interprétation de la part des navigateurs, ce qui explique qu'une même page web puisse s'afficher différemment d'un navigateur Internet à l'autre. De plus, il arrive parfois que certains éditeurs de logiciels ajoutent des instructions HTML propriétaires, c'est-à-dire ne faisant pas partie des spécifications du W3C. Ainsi des pages web contenant ce type d'instruction pourront être parfaitement affichées sur un navigateur et totalement illisibles sur les autres, d'où la nécessité de créer des pages web respectant les recommandations du W3C afin de permettre leur consultation par le plus grand nombre.

Notre première page web

Le HTML n'est pas un langage de programmation, Une page HTML est donc un simple fichier texte contenant des balises (*tags*) permettant de mettre en forme le texte, les images, etc. Une balise est une commande (un nom) encadrée par le caractère inférieur (<) et le caractère supérieur (>) par exemple "<H1>". Les balises HTML sont généralement présentes par paire afin d'agir sur les éléments qu'elles encadrent. La première est appelée "balise d'ouverture" (parfois balise ouvrante) et la seconde "balise de fermeture" (ou fermante). La balise fermante est précédé du caractère / :

```
<marqueur> Votre texte formaté </marqueur>
```

Ainsi les balises et permettent de mettre en gras le texte qu'elles encadrent. Les balises HTML peuvent parfois être uniques : la balise
 représente par exemple un retour à la ligne. Les balises ne sont pas sensibles à la casse, c'est-à-dire qu'elles peuvent être indifféremment en minuscules ou en majuscules.

Un attribut est un élément, présent au sein de la balise ouvrante, permettant de définir des propriétés supplémentaires à la balise à laquelle il appartient. Les attributs se présentent la plupart du temps comme une paire clé-valeur, sous la forme cle="valeur", mais certains attributs sont parfois une seule clé. Voici un exemple d'attribut pour la balise <p> (balise définissant un paragraphe) permettant de spécifier que le texte doit être aligné sur la droite :

```
<p align="right">Exemple de paragraphe</p>
```

Chaque balise peut comporter un ou plusieurs attributs, chacun pouvant avoir (aucune,) une ou plusieurs valeurs.

Notre première page web se sépare en deux éléments : un entête (*header*) et un corps (*body*). Si vous souhaitez apprendre plus de balises (listes, tables, etc.), vous pouvez aller voir ce tutoriel :

 Tutoriel HTML

- <http://asl.univ-montp3.fr/tutorielHTML/>

1.5.2 Mises en forme avancées

L'utilisation des tables imbriquées

Le web évoluant et prenant de plus en plus d'importance, les webmestres ont cherché à présenter l'information de façon percutante. On demandait des mises en pages visuelles complexes. Mais la norme HTML 3 n'était pas bien équipée pour cela : elle contenait encore peu de balises pour cela. Une des solutions des pionniers a été l'utilisation des balises <table imbriquées les unes dans les autres pour présenter une structure à l'écran. Centrer une page ayant une certaine largeur ? Une table centrée dans la cellule d'une table qui remplit l'écran.

Les frames

Le système des *frames* apparu avec la norme HTML 4 permet d'afficher simultanément plusieurs pages à l'écran.

1.5.3 Les feuilles de style

Présentation de la norme CSS

Les feuilles de style CSS permettent une meilleure gestion des pages HTML. Elles séparent le contenu de la forme et offrent une syntaxe simple pour définir le format. Bref, elles agissent à la manière des styles dans Word.

En effet immédiat, elles allègent la taille des documents publiés. Imaginons que tous les titres doivent être mis en vert, gras, taille 14. Plutôt que d'avoir les balises de format répétées (`...`) à chaque fois,

L'utilisation de feuilles de styles permet aussi de redéfinir la charte graphique d'un site très rapidement. L'information de la charte graphique est en effet centralisée dans un ou deux fichiers seulement.

Utilisation avancée : plusieurs médias

Les feuilles de style CSS introduisent de plus la notion de média. Le consortium WWW définit les médias "screen", "print" et "audio". Le navigateur Opéra a pris l'initiative d'y ajouter un média "projection". Pour chacun de ces médias, le webmestre peut spécifier des règles différentes, ce qui permet d'afficher des pages visuellement très attrayantes (utilisation d'images, etc) qui s'imprimeront toutefois sobrement.

1.6 Les langages balisés

1.6.1 Retour aux sources : le SGML

Le Langage Standard Généralisé de Balisage (Standard Generalized Markup Language ou SGML, défini par le document [ISO8879]), est un métalangage, un formalisme qui permet de définir des langages de balisage. Chaque langage de balisage est une *grammaire* de balises que les documents doivent respecter. HTML est ainsi une grammaire SGML. Si l'information n'est pas gardée sous une forme correspondant à celle d'un document imprimé particulier (séquentiellement) mais est représentée par sa structure logique, on bénéficie de l'"usabilité" et de la "réusabilité" de l'information :

- mise à jour
- extraction d'information
- publication sous différentes formes

Ainsi, entre autres, un document structuré pourra (plus ou moins) aisément être converti vers un format séquentiel, le contraire n'étant pas vrai.

L'origine de SGML remonte à 1969, où Charles F. Goldfarb travaille chez IBM sur le GML. De nos jours, SGML est encore utilisé par beaucoup de sociétés :

- AAP (*American Association of Publishers*)
- Manuels techniques (IBM, HP, O'Reilly...)
- Industrie aéronautique (Airbus, Boeing, Lufthansa...)
- Agences de presse
- Publication Internet
- ...

Structure du document : la DTD

La structure du document est spécifiée *via* un document SGML appelé DTD (Document Tree Definition). Le fichier DTD énonce les règles générales qui régissent un type de document :

- le nom des éléments qui peuvent être utilisés (article, auteur,...)
- le contenu de chaque élément
- le nombre d'occurrences et l'ordre de chaque éléments
- si une balise de fin ou de début peut être omise
- les attributs (propriétés) éventuels d'une balise
- les entités qui peuvent être utilisées

TAB. 1.3: Opérateurs et ordre des contenus dans une DTD

,	tous les éléments doivent apparaître dans l'ordre donné
&	tous doivent apparaître, ordre quelconque
	un et un seul doit apparaître
+	élément obligatoire et répétable (1 fois ou plus)
?	élément optionnel (0 ou 1 fois)
	élément optionnel et répétable (0 fois ou plus)

L'écriture d'un document SGML s'effectue donc en deux temps : l'écriture (ou le choix) d'une DTD puis l'écriture du contenu conformément à la DTD choisie.

Par exemple, la DTD fixée pour HTML comprend la définition suivante :

```
<!ELEMENT HTML O O (HEAD, BODY)>
```

Cette déclaration indique qu'un document HTML contient un entête **suivi** d'un corps. La partie intermédiaire permet la minimisation pour la balise HTML (O pour Omise). Les opérateurs suivants permettent de fixer le comportement des contenus :

Encore un exemple : la définition HTML de la balise <DL> (*Definition List*) :

```
<!ELEMENT DL -- (DT*, DD ?)+>
```

qui indique qu'une liste de définitions doit satisfaire les contraintes suivantes :

- les balises <DL> et <DT> doivent toujours être présentes
- elle contient une ou plusieurs occurrences (...)+
- de 0 ou plus de balises <DT> (DT*)
- qui peuvent être suivies (,)
- d'au plus une balise <DD>

Un élément peut contenir un autre élément (une table contient des lignes qui contiennent des cellules) ou des caractères. Le contenu de chaque élément doit appartenir à l'un des éléments suivants :

PCDATA Données textuelles analysées (balises ou entités interdites). Exemple :

```
<!ELEMENT TITLE -- (#PCDATA)>
```

RCDATA Données textuelles remplaçables

CDATA Données textuelles

ANY PCDATA ou tout autre élément

EMPTY Contenu vide

Enfin, chaque élément (balise) peut avoir des attributs spécifiés la plupart du temps sous la forme "*attribut=valeur*". Exemple :

```
<lettre type="business" from="Eric">
```

La DTD HTML

Ainsi, la norme HTML n'est rien d'autre qu'une grammaire SGML. Son succès s'explique surtout par la pauvreté et le laxisme de cette grammaire : la DTD est en effet peu contraignante. Par exemple, une balise de section <H2> peut être placée à l'intérieur d'un paragraphe ou pour baliser un paragraphe. Chaque navigateur interprète la norme à sa façon. Le fait est que HTML n'est pas une norme mondiale, mais une norme édictée par le consortium WWW, union d'or


Pour plus d'information sur le SGML, son utilisation et l'écriture de DTD on se référera à Herwijnen (1995)

1.6.2 XML

La présentation du SGML était longue. Elle est très utile pour présenter simplement son dérivé : le XML (*Extensible Markup Language*). Le but premier de XML était de rendre accessible l'utilisation de SGML pour le *World Wide Web*, en bénéficiant ainsi du côté extensible, que ne possède pas la norme HTML figée. Ainsi, XML facilite(ra) la définition de types de documents génériques qu'il sera facile de créer, gérer, afficher et partager via le web..

D'un point de vue informatique, XML est un dialecte simplifié de SGML, dont les parties plus complexes et les moins utilisées ont été supprimées. Une spécificité importante est que XML est plus *contraignant* que SGML : la norme SGML autorise, par exemple, des balises non fermées, ce que ne fait pas XML. L'ensemble des simplifications (des contraintes ajoutées) font qu'il est possible de valider un document XML sans connaître sa DTD. Par ailleurs, la production (automatique ou non) de documents XML est elle aussi rendue plus facile. Un ajout à SGML est le support Unicode, indispensable pour une adoption globale du système. Enfin, un document XML pourra toujours être considéré comme un document SGML.

Il existe déjà des éditeurs XML commerciaux WYSIWYG. Pour une liste (non exhaustive), on pourra consulter :

 Editeurs XML

– <http://www.xmlsoftware.com/editors.html>

XSLT

XSLT est une grammaire XML permettant de décrire des transformations à appliquer sur un document XML. Son application principale vise à transformer un document XML vers un autre document XML mais il peut aussi mener à un document d'un type quelconque (y compris binaire).

Une des principales applications est la transformation d'un document XML en HTML. La version xHTML de ce document fait appel à une feuille XSLT pour ajouter les acronymes.

XSLT est assez technique. Cette norme s'appuie de plus sur d'autres normes. La norme XPath fournit un mécanisme de navigation dans la structure arborescente du document XML (sélection de nœud(s) ou d'attribut(s) répondant à une certaine logique).

XPath

XPath est une norme permettant d'interroger des fichiers XML et d'accéder à certains éléments. La syntaxe permet des accès directs aux nœuds du fichier selon : l'emplacement dans l'arbre (parents, profondeur,...), la valeur de propriétés, le contenu ou toute combinaison de l'ensemble. Un jeu de fonctions ajoute des manipulations basiques (calculs, booléens, chaînes de caractères).

XPath a peu d'utilité seul. Il est en revanche indispensable à XSLT qui connaît ce dialecte. Ainsi, XPath sélectionne des éléments et XSLT définit les manipulations à effectuer sur ces éléments.

La syntaxe XPath suivante permet de récupérer tous les titres de la version XML de ce support :

```
/book/mainmatter//heading/text()
```

La syntaxe XPath suivante permet de lister tous les paragraphes inclus dans un autre paragraphe. Cet emboîtement n'est en effet pas acceptable par la DTD DocBook.

```
//p/p
```

XLS-FO

XLS-FO (*eXtended Stylesheet Language - Formatting Object*) est le vocabulaire qui décrit les mises en forme de documents XML quelque soit le support : écran, papier, mais aussi dispositifs audio ou vidéo. XLS-FO est à XML ce que DVI ou PDF sont à L^AT_EX. L'objectif (fort ambitieux) de XLS-FO est de créer un arbre d'aires où une aire est une zone d'affichage (visuelle ou auditive). Les aires sont de 2 types :

- les aires de blocs (*block element*) s'empilent les unes sur les autres
- Les aires en-ligne (*inline element*) s'empilent les unes à côté des autres.

XLS-FO fournit l'ensemble des commandes de contrôle de chaque aire : présentation du contenu, direction de l'empilement (écriture de gauche à droite ou inversement, de haut en bas...)...

A ma connaissance, seul Apache développe un moteur suivant cette recommandation (FOP : Formatting Objects Processor) en donnant la priorité au format cible PDF.

 Moteur FOP d'Apache


– <http://xml.apache.org/fop/>

TAB. 1.4: Quelques dialectes XML

Dialecte	Objectif
CRT	Case Report Tabulation Data (essais cliniques)
hrXML	Langage de représentation de CV (pour les ressources humaines)
MathML	Langage de représentation de contenu mathématique
MusicXML	Langage de représentation de contenu musical (partitions)
SVG	Langage de représentation graphique vectorielle
UIML, XAML, XUL, Flex	Langages de représentation d'interfaces graphiques
VoiceXML	Langage de représentation de contenu vocal
XHTML	Langage de représentation de document hypertexte
XSLT	Langage de transformation XML
XSL-FO	Langage de représentation de document
XSLT	Langage de transformation XML

1.6.3 Quelques DTD

La table 1.4 présente quelques dialectes (ou grammaires) XML spécifiques à certaines utilisations. La liste n'est pas exhaustive du tout : pour le SGML ont été disponibles des centaines de DTD...


 Liste de DTD XML maintenu sur le site XML.org
 – <http://www.xml.org/xml/registry.jsp>

xHTML

xHTML est l'évolution de HTML. Là où HTML est une application figée et structurellement impropre de SGML, xHTML est la déclinaison sémantique à la XML. En fait, il y a peu de changements. On notera l'abandon de beaucoup de balises de format, comme `` pour le gras ou `<i>` pour l'italique. En effet, la philosophie SGML/XML exige la séparation totale de la forme et du contenu

Un des avantages de travailler avec xHTML est la facilité d'inclure des morceaux XML régis par d'autres DTD, comme MathML ou SVG. Les navigateurs modernes comprennent en effet ces formats et l'interprétation (l'affichage) est assuré sans plus de travail.

Enfin, des directives européennes vont obliger les organismes officiels (gouvernements, etc.) à ne présenter que du contenu en xHTML. Seuls des documents validés (syntaxiquement corrects) peuvent être affichés. Un des enjeux cruciaux derrière ces directives est l'accessibilité. En effet, des moteurs XSLT sont en développement pour assurer des actions telles que conversion de contenu en braille ou diction vocale.

 Articles d'alsacreations sur la création de sites web "aux standards"
 – <http://www.alsacreations.com/articles/>

MathML

MathML est une DTD dont la finalité est l'encodage de formules mathématiques. La norme est promise à un bel avenir. Actuellement, certains navigateurs gèrent l'affichage du MathML en natif (c'est le cas du moteur Mozilla - Firefox). Pour les autres (Explorer), des *plug-in* existent (*MathPlayer*).

Un des problèmes de la norme MathML est inhérent à la spécification XML : la nécessité d'aligner un nombre conséquent de balises. La célèbre formule d'équivalence masse / énergie d'Einstein s'écrira :


```
<math>
  <mi>E</mi>
  <mo>=</mo>
  <mi>m</mi>
```

```

<msup>
  <mi>c</mi>
  <mn>2</mn>
</msup>
</math>

```

La même équation en $\text{T}_{\text{E}}\text{X}$ s'écrit (bien plus naturellement) $E=mc^2$... Un des points importants est que le XML est rarement censé être écrit directement par un opérateur humain. On s'attend généralement à ce qu'il soit généré. Certains éditeurs WYSIWYG exportent nativement en MathML. À noter aussi l'existence d'outils permettant la conversion $\text{T}_{\text{E}}\text{X} \rightarrow$ MathML ou réciproquement. On notera particulièrement le javascript `AsciiMathML` qui transforme à la volée sur un poste client du code $\text{T}_{\text{E}}\text{X}$ en MathML. On trouvera une discussion intéressante sur l'usage du MathML à l'adresse suivante :

 [MathML Workflows in STM Publishing](http://www.dessci.com/en/reference/white_papers/mathml_workflows.htm)

– http://www.dessci.com/en/reference/white_papers/mathml_workflows.htm

DocBook

DocBook est une DTD ancienne puisqu'elle existait déjà en SGML.

Sa vocation principale est l'écriture de documentation (technique ou non) ainsi que de manuels. Forte de ses quelques 200 balises différentes, la norme DocBook peut à peu près couvrir tous les besoins d'écriture de documents. De nombreuses feuilles de styles existent permettant la conversion en RTF, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, xHTML... Reste qu'un nombre si important de balises (environnements) peut paraître inquiétant pour un novice.

Tbook

Tbook est une DTD créée par un particulier pour l'écriture de sa thèse. Il trouvait que DocBook était d'une part trop *vaste* pour l'écriture de documents scientifiques et mal adapté à l'écriture de formules mathématiques (sauf grands bidouillages). La DTD Tbook ne comprend qu'une quarantaine de balises, suffisantes pour la majorité des besoins.

Le site officiel de Tbook propose le téléchargement de l'ensemble des outils nécessaires pour une bonne utilisation : les DTD mais aussi des feuilles de styles permettant l'exportation en $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, xHTML et DocBook. Plus intéressant encore, des scripts se chargent de la gestion des images (format JPEG pour le web, Postscript pour $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ et PDF pour l'utilisation de pdf $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$).

 [Le site officiel de Tbook](http://tbookdtd.sourceforge.net/)

– <http://tbookdtd.sourceforge.net/>

L'utilisation d'un encodage Unicode comme vu précédemment allège fortement la tâche de la saisie d'équations : Unicode définit en effet la majorité des symboles mathématiques nécessaires (intégrale, somme, limites, etc.). Une équation mathématique pourra alors être encodée en MathML directement avec des symboles Unicode, ajoutant la visualisation directe lors de l'édition en mode texte.

```

<m>
x_{\text{eff}} \neq \left( \int_0^\infty \sin(\tilde{x}) \frac{\sqrt[3]{1/e}}{\beta} dx \right) \neq \lim_{x \rightarrow \infty} \frac{1}{x}
</m>

```

Sera affiché


$$x_{\text{eff}} \neq \left(\int_0^\infty \sin(\tilde{x}) \frac{\sqrt[3]{1/e}}{\beta} dx \right) \neq \lim_{x \rightarrow \infty} \frac{1}{x}$$

Les caractères Unicode (tel que "≠") ne sont disponibles qu'en encodage UTF. Si vous encodez en ASCII, il faudra utiliser le nom de l'entité `&neq` ;

SVG

Le format SVG est un dialecte XML orienté vers la description de contenu graphique. Vectoriel de base, la norme prévoit une multitude de fonctionnalités très intéressante : superposition d'éléments, gestion de la transparence, animations,...

Peu de navigateurs ont intégré nativement des outils pour l'affichage de SVG. On pourra toutefois utiliser un plug-in comme celui développé par Adobe.


 Adobe SVG viewer version 3

– <http://www.adobe.com/svg/viewer/install/main.html>

Peu de logiciels créent des fichiers au format SVG. Toutefois, ce format est bien traité à l'importation-exportation (en utilisant, par exemple, le couteau suisse de l'image ImageMagick). Par conséquent, si l'on peut créer un fichier SVG, ce format peut être un bon choix. L'environnement R dispose du package `RSvgDevice` ajoutant un *device* SVG.

1.6.4 Quelques démonstrations supplémentaires


ActiveMath

 ActiveMath

– <http://www.activemath.org>

SciWriter

Logiciel : SciWriter -> Démonstration XML+MathML

 Soft4Science SciWriter

– <http://www.soft4science.com/>

Chapitre 2

Quelques (bonnes) pratiques

2.1 De la sémantisation

2.1.1 Séparation forme/contenu

On l'a vu avec l'évolution du HTML : la séparation de la forme et du contenu est une bonne chose. L'aspect visuel peut être modifié aisément, sans devoir retoucher au contenu et la gestion du contenu est elle aussi simplifiée (puisqu'on n'y mélange pas d'instructions de format).

2.1.2 Structure sémantique

Pour atteindre le but d'une bonne séparation forme/contenu, on passe par une approche de sémantisation du texte. Les bonnes questions sont :

- Pourquoi mettre ce mot en gras ?
- Pourquoi mettre ce *groupe de mots* en gras ?
- Quel sens a ce bloc : titre ? note de bas de page ? contenu d'une cellule ?

Répondre à toutes ces questions nécessite un effort supplémentaire. Le prix à payer est cependant vite rentabilisé. Un document structuré offre des possibilités supplémentaires : accéder par exemple à tous les titres de niveau 3. Modifier l'apparence des premières lettres de chaque paragraphe automatiquement.

2.1.3 Métadonnées

La sémantisation revient à fournir de l'information sur l'information (voici un paragraphe, mais pas n'importe quel paragraphe — c'est un paragraphe d'introduction). On parle alors souvent de métadonnées (*meta-data*). Les métadonnées sont présentes depuis le début des pages HTML (titre, mots-clés). Word propose aussi depuis longtemps l'ajout de métadonnées (auteur, date, etc.). L'utilisation des métadonnées est, elle, assez récente (principalement avec l'émergence des moteurs de recherche). Les métadonnées, aussi considérées comme structures, sont essentielles au statisticien. Le système SAS en emploie depuis longtemps et en ajoute de plus en plus¹.

2.2 Liens entre les formats

Depuis longtemps, l'enjeu du multi-formats est important. Depuis le WWW, il l'est encore plus souvent : beaucoup de documents doivent être imprimés *et* publiés sur Internet, soit deux médias différents.

C'est la raison pour laquelle de nombreux outils de conversion existent. Passer de L^AT_EX à HTML (ou dans l'autre sens), de L^AT_EX au format RTF (pour lecture dans Word)...


De façons basique, on distinguera deux grandes catégories d'approches :

¹ Les datasets sont constituées de deux parties : description et données. Avec la version 9 de la plateforme, tout un ensemble côté serveur permet la gestion de métadonnées, incluant - entre autres - le suivi des modifications.


L'approche brute La grande majorité des convertisseurs ont une approche assez brute : la conversion des commandes d'un format vers les commandes d'un autre. La tâche est souvent ardue et il est souvent difficile de faire coïncider toutes les commandes. Ce qui fait que la majorité des convertisseurs directs viennent avec des avertissements de type : *Ne reconnaît que les fonctionnalités x, y, z du format source/destination.*

Les chaînes directes L'approche SGML/XML permet des chaînes directes via XSLT et XSL-FO. Là encore, toutes les fonctionnalités ne sont pas forcément prises en compte. L'avantage est cependant que la conversion passe par des feuilles de style que l'utilisateur peut étendre (à condition d'apprendre des rudiments de XSLT).


Pour une liste (non exhaustive) d'outils de conversion $\LaTeX \leftrightarrow \text{format}$, on pourra consulter le site suivant :

 Liste de convertisseurs de er vers \LaTeX
– <http://www.tug.org/utilities/texconv/>

On retiendra particulièrement TeX4ht qui est le seul à générer du xHTML propre. Les équations sont converties en MathML.

 TeX4ht : un très bon convertisseur \LaTeX vers xHTML
– <http://www.cse.ohio-state.edu/~gurari/TeX4ht/>

Le projet Hermès semble aussi très prometteur (uniquement sous Unix pour l'instant)

 Projet Hermès : outil d'e-publication sémantique pour \LaTeX
– <http://www.aei.mpg.de/hermes/>


2.3 Literate Programming


Le *Literate Programming*, c'est encore un concept et une application de Donald E. Knuth, le créateur de \TeX . Sa volonté : une meilleure documentation d'un code source de programmation. Son objectif : élever l'art de la programmation à l'art de l'écriture, d'où le nom. L'idée est que du code de programmation ne doit pas seulement être "lu" et interprété par l'ordinateur (le compilateur), mais aussi par tout lecteur. Un programme pourrait ainsi contenir sa propre documentation et permettre à tout lecteur de le comprendre. La programmation vue par D. Knuth lie code et documentation sous une forme compréhensible par le lecteur humain. Différents systèmes existent (cweb, noweb, funnelweb, etc.). Une solution permettant d'embarquer du code S dans un fichier \LaTeX ou HTML existe pour R (voir la section 3.2.1).

Pratiquement, on écrit un unique fichier source mixant code et texte. Puis, sur ce fichier source, on peut appliquer deux opérations :

Tangling Cette action consiste à parser le document source pour en extraire uniquement le code du programme. Cette étape pourra être réalisée juste avant compilation / exécution.

Weaving Cette action consiste à parser le document source pour créer la documentation du programme dans un format acceptable pour le lecteur (\LaTeX ou encore HTML...).

 Présentation du concept *Literate Programming* par Knuth
– <http://tex.loria.fr/litte/knuthweb.pdf>

 Le système noweb
– <http://www.eecs.harvard.edu/~nr/noweb/>

2.4 Et Word dans tout ça ?

Le grand absent dans cette discussion... Word est un traitement de texte WYSIWYG (What You See Is What You Get - prononcer Ouizi-ouigue). En fait, pour toutes les raisons évoquées précédemment, Word est loin d'être le traitement de texte idéal. Pour des documents courts à usage unique, il peut faire l'affaire. Mais mieux vaut y réfléchir à deux fois avant de l'utiliser.

Certaines manières d'utiliser Word rendent la vie plus facile. Ainsi, l'utilisation des styles *partout* revient à dissocier forme et contenu. Encore faut-il s'astreindre à créer des styles pour tout au lieu de cliquer directement sur "centrer" ou "gras" et à les utiliser, même pour mettre des termes en italique.

En janvier 2005, Microsoft a annoncé l'ouverture du format Word 2003 et autorise (enfin ?) les développeurs à générer des fichiers compatibles depuis leurs applications. Reste que cela n'autorise que le format XML interne, non compressé (donc des fichiers qui occupent beaucoup de place). Enfin, l'utilisation est soumise à une licence qui semble extrêmement restrictive : l'inclusion du support Word 2003 empêcherait le dépôt de propriété intellectuelle.

2.5 DOM et API


Les normes XPath et XSLT permettent de manipuler des documents XML, ce qui est fort pratique. Toutefois, il n'y a pas que les documents XML que l'on peut vouloir traiter de manière automatique. L'activité de traitement automatique est souvent importante : remplacer le nom d'une entreprise dans tous les documents, ajouter des morceaux de texte, extraire tous les résumés, compter les mots, récupérer toutes les images sont quelques traitements que l'on aimerait pouvoir accomplir aisément.

Le *Document Object Model* (DOM) définit la façon dont on peut accéder aux parties d'un document. La structure arborescente dote SGML et XML d'un DOM natif (ce qu'exploite XPath). C'est clair : quand on vient à la manipulation d'information, rien ne vaut ces formats.

Toutefois, d'autres formats sont aussi dotés d'un DOM. C'est le cas du format DOC de Word. Tout n'est pas possible dans Word : on ne peut pas, par exemple, inclure une image dans une équation.

Une différence pourtant, et de taille : peu d'outils peuvent exploiter le DOM de Word. En effet, qui dit traitement *automatique* dit *programmation*. La majorité des langages de programmation peuvent adresser du XML. Microsoft fournit avec Word des API (*Application Programming Interface*) permettant les manipulations. Le langage roi pour l'utilisation de ces API reste tout de même Visual Basic. Notons que les API ne sont utilisables que si l'ordinateur dispose d'un Word installé...

Pour les personnes qui devraient manipuler beaucoup de fichier L^AT_EX, il existe un projet d'API en Perl pour manipuler un pseudo-DOM :

 L^AT_EX TOM - API Perl pour manipuler des documents L^AT_EX
- http://br.endernet.org/~akrowne/elaine/latex_tom/

Chapitre 3

Pour les statisticiens

3.1 Reporting direct

3.1.1 SAS

\SAS dispose de l'environnement leader en reporting, grâce à son système ODS (Output Delivery System). Le système consiste en une série de commandes SAS permettant la gestion des sorties :

- Exportation vers différents formats : HTML, Postscript, PDF, RTF, L^AT_EX (version 9), DocBook, XML (SAS DTD), ...
- Sélection du contenu à exporter
- Manipulation de la structure et du format des tables
- Création de formats d'exportation customisés
- Récupération de toute table sortie dans un dataset


Instructions ODS...

Les différentes instructions suivantes contrôlent le système ODS :

- Choisir une destination
 - ODS LISTING ...
 - ODS HTML ...
 - ODS OUTPUT ...
- Sélectionner les objets à créer
 - ODS SELECT ...
 - ODS EXCLUDE ...
- Contrôler les messages affichés dans le Log
 - ODS SHOW ...
 - ODS TRACE ...
 - ODS VERIFY ...
- Sélectionner un chemin pour les *templates* personnalisés
 - ODS PATH ...

Langages à balises - tagset

Depuis la version 9, un mécanisme très générique a été mis en place pour l'exportation vers des langages balisés. En fait, il est possible de définir assez facilement des nouveaux *drivers* pour exporter vers n'importe quel format. Si la pensée initiale était sûrement de laisser la liberté de générer du XML selon n'importe quelle DTD, des utilisateurs ont profité de cette possibilité pour ajouter des exportations diverses : SQL ou encore Excel direct en sont deux exemples. On consultera :

 Informations sur les *tagsets* disponibles pour SAS ODS

– <http://support.sas.com/rnd/base/index-ods-resources.html>

PROC DOCUMENT

Depuis la version 8, la procédure `PROC DOCUMENT` permet de sauvegarder un contenu. L'exportation peut ainsi être réalisée ultérieurement, ou sur un autre ordinateur, avec les règles voulues. De plus, un même document pourra être exporté de multiples fois sans devoir pour autant réaliser à nouveau les calculs intermédiaires nécessaires menant à son contenu.

3.1.2 SPSS

SPSS a introduit dans sa version 11 le système OMS (*Output Management System*) qui ressemble fort au système ODS de SAS. Je n'ai pas encore beaucoup investigué.

3.1.3 Splus SPXML

SPXML est une librairie fournie en standard avec SPlus 6 permettant d'exporter en XML les objets de classes suivantes : `data frame`, `list`, `array`, `vector`, `matrix`, `function`, `call`.

Malheureusement, toute modification nécessitera des connaissances en XSLT. Par ailleurs, le moteur utilisé n'est pas des plus rapides¹.

Notons la possibilité d'importer des datasets SAS exportés en XML.

3.1.4 R2HTML

R2HTML est un package d'exportation en HTML que j'ai écrit et que je maintiens activement depuis plusieurs mois. L'approche en est très naïve mais donne des résultats : des méthodes d'exportation sont définies pour la majorité des objets R. L'exportation est réalisée brutalement avec des `cat()`.

Il existe trois façons d'utiliser les fonctions du package :

- Création de méthodes `HTML.maclasse` basé sur les méthodes existantes pour réaliser des rapports personnalisés
- Utilisation en direct au cours d'une session *via* `HTMLStart()` et `HTMLStop()`
- Utilisation conjuguée avec le système Sweave

Un article publié dans Rnews décrit plus en détail ces trois fonctionnements. On le trouvera à :

 R2HTML - Démonstrations et documentation

– <http://www.stat.ucl.ac.be/R2HTML>

3.1.5 Projet ROMA

ROMA pour *R Output MAnager*. Derrière ce nom plus si original se cache un système "à la ODS" pour R. C'est ce projet qui est à la base de mes investigations dans les formats et outils pour le *reporting*. Une version de démonstration du concept existe, mais je dois tout reprendre à zéro, principalement pour des raisons d'efficacité des classes S4. Les buts à atteindre sont les suivants :

- Un DOM interne à R
- Des API pour créer et manipuler des objets de type rapport
- Des drivers d'exportation. Les formats prioritaires sont XML, xHTML et \LaTeX .
- La possibilité de définir ses propres drivers d'exportation (langages balisés)
- La possibilité de définir ses propres styles depuis R
- Un support de l'international ("," *versus* "."), par exemple)

¹ Il faut dire que tout est en Java, stocké sur un serveur distant. Il faut donc commencer par télécharger le code Java sur le poste.

3.2 Rapports dynamiques

3.2.1 Sweave

Sweave est l'adaptation du système de *literate programming* de Knuth pour l'environnement R. On écrit donc un fichier source en \LaTeX incluant du code R. Sweave s'occupe après de tout : exécution du code R, création des graphiques en Postscript et PDF (respectivement pour \LaTeX et $\pdf\LaTeX$) et inclusion des figures.

Le package R2HTML ajoute un driver pour utiliser Sweave avec des pages HTML.

Deux packages rendent encore plus agréable l'utilisation de Sweave avec \LaTeX :


xtable permet d'exporter directement en \LaTeX certains objets R (surtout création de tables).

Hmisc comporte différentes fonctions écrite par Frank Harrell.

De manière générale, toute personne intéressée par la production de sorties de qualités depuis R se tournera pour l'instant vers \LaTeX . La lecture de *Statistical Tables and Plots using S and \LaTeX* de Frank Harrell est un *must*.

 Sweave, de Friedrich Leisch


– <http://www.ci.tuwien.ac.at/~leisch/Sweave/>

 Statistical Tables and Plots using S and \LaTeX

– <http://lib.stat.cmu.edu/S/Harrell/doc/summary.pdf>

3.2.2 Rpad

Rpad, développé par Tom Short, est défini comme un programme d'analyses statistiques dynamiques, basés sur le web. La version originelle réside sur un serveur et fonctionne avec des pages HTML comprenant du code . L'utilisateur peut à tout moment soumettre la page, dont le code sera évalué par le serveur. Le retour (assez rapide) est une nouvelle page web dont le code aura été remplacé par les sorties (mises en forme à la volée *via* le package R2HTML).

 Rpad home page

– <http://www.rpad.org>

Le principe est assez proche d'un système de *literate programming*. Toutefois, l'utilisation de pages HTML ajoute un plus indéniable : la facilité à construire des interfaces graphiques.

Une version locale (ne nécessitant pas de serveur) de Rpad est en développement et devrait être rendue publique d'ici le mois de mars 2005. Une telle version permettrait le développement rapide d'applications avec interfaces graphiques multi-plateformes pour R.

3.2.3 SAS Office

SAS a lancé un produit autorisant la communication entre un produit Microsoft Office et un serveur SAS. Le but premier est la création de rapports automatisée pour les employés : on lance Word (ou Excel), on clique sur un élément de menu et le rapport arrive. Ce produit n'est pas disponible à l'UCL. Et pour cause : il s'appuie sur le mécanisme de tâches enregistrées, lesquelles nécessitent une version spécifique de serveur SAS. Bref, le produit est très ciblé et serait en son état de peu d'utilité aux universitaires. Nul doute, toutefois, qu'il évolue vers un produit plus polyvalent et plus proche d'un véritable système de *literate programming* pour Office.

3.3 Pour les données ?

La problématique est similaire : il existe beaucoup de formats différents : Texte (CSV par exemple), Excel XLS, SAS datasets, SPSS SAV, etc. Il est possible de distinguer deux usages différents : la présentation de


TAB. 3.1: Capacités à représenter les structures des données statistiques

	R/SPlus	MatLab	Octave	Excel	SPSS	SAS	Minitab
$\pm\infty$, NaN	X	X	X				
Données manquantes	X		X	X	X	X	X
Variable : booléenne	X	X		X	X	X	X
Variable : nominale	X	strings	strings	strings	X	X	strings
Variable : ordinale	X				X	X	X
Donnée : integer	X	X				X	
Donnée : real	X	X	X	X	X	X	X
Donnée : complex	X	X	X				
Donnée : string	X	X	X	X	X	X	X
Donnée : date/time	X	X	X	X	X	X	X
Structure : matrice	X	X	X	X	X	X	X
Structure : array	X	X	X				X
Structure : list	X	X	X				
meta-data	X			X	X		

données statistiques et la description (structure) en vue de communication (importations/exportations dans les différents logiciels).

3.3.1 Structuration des données

Les données statistiques sont souvent pensées en terme de table plate à deux dimensions (le "classique" : individus en lignes, variables en colonnes). C'est une erreur. Les données statistiques ont besoin de bien plus de structure. Comment savoir, par exemple, si une table représente vraiment le croisement (individus \times variables) ou le croisement de deux variables (table de contingence) ? Est-il légitime de représenter des données spatiales ou temporelles sous une forme tabulaire ?

 Les fichiers de données en statistique

– <http://www.stat.ucl.ac.be/ISpersonnel/lecoutre/stats/fichiers/data.pdf>

Par ailleurs, toutes les données n'ont pas la même signification. Ainsi, on doit pouvoir manipuler des données entières, réelles ou encore complexes, mais aussi l'infini, l'indéfini et les données manquantes.

Enfin, les variables elle aussi peuvent être de différentes natures : nominales, ordinales ou continues.

A titre informatif, voici la table 3.1 recense une liste de fonctionnalités souhaitables implémentées ou non dans divers logiciels orientés vers le calcul scientifique.

StatDataML

est une DTD dédiée à la structuration de données statistiques, permettant les échanges entre logiciels. La structure emprunte beaucoup au langage S (aspect récursif) tout en ajoutant des méta-données. L'ensemble couvre toutes les capacités requises de la table 3.1. Des fonctions existent pour R, SPlus, MatLab et SAS pour l'importation / exportation de données suivant ce format.

Bibliographie

HERWIJNEN, Eric Van, 1995 : *SGML pratique*. Seconde éd. International Thomson Publishing. 14

Index

A

API 21
ASCII 5

B

balise 11*ff.*, 23
body 12

C

Convertisseur 19
CSS 13

D

DocBook voir Format|DocBook
DOM (*Document Object Model*) 21
DTD 13, 16

E

EPS (*Encapsulated Postscript*) voir Format|EPS

F

FOP 15
Format
– DocBook 17
– EPS 8
– HTML 11, 14, 24
– L^AT_EX 17, 25
– LyX 11
– PDF 8
– RTF 10
– SGML 13
– SVG 18
– tbook 17
– T_EX 17
– TXT 6
– xHTML 16
– XML 14

H

header 12
Herwijnen (1995) 14
HTML voir Format|HTML

L

L^AT_EX voir Format|TeX
Logiciel
– R 20, 25
– Rpad 25

– SAS 8, 19, 23*f.*
– SciWriter 18
– SPlus 24
– SPSS 24
– Word 21
LyX voir Format|LyX

M

MathML 16*f.*
média 13

O

ODS 23
OMS (*Output Management System*) 24
Outil
– Adobe SVG viewer 18
– ImageMagick 18
– R 24
– Sweave 25

P

PDF voir Format|PDF
Postscript voir Format|EPS

R

R2HTML 24
RTF (*Rich Text Format*) voir Format|RTF

S

SGML voir Format|SGML
SPXML 24
SQL 23
StatDataML 26
SVG voir Format|SVG

T

Tangling 20
tbook voir Format|tbook
T_EX voir Format|TeX
Texte voir Format|TXT
– Editeur 7

U

Unicode 6, 17

W

Weaving 20

X

xHTML voir Format|xHTML
XLS-FO 15

XML voir Format|XML
XPath 15
Xpath 21
XSLT 15, 20*f.*