

Ateliers d'initiation à L^AT_EX- Troisième séance

Jérôme Balthazar

Pierre-Edouard Jacoby

Martin Van den Abbeele

Laurent Bataille

Version mars 2021

Objectifs de la séance

- Prise en main des tableaux et des matrices
- Utilisation d'outils de mise en forme automatique
- Gestion de projets en LaTeX : comment structurer de gros documents.

1 Les tableaux et les matrices

Les matrices constituent la transition la plus directe entre les environnements mathématiques et les tableaux.

Code minimaliste décrivant une matrice - [↗](#)

```
\begin{equation*}
\mathbf{\sigma} =
\begin{matrix}
\sigma_{11} & \sigma_{12} & \sigma_{13} \\
\sigma_{21} & \sigma_{22} & \sigma_{23} \\
\sigma_{31} & \sigma_{32} & \sigma_{33}
\end{matrix}
\end{equation*}
```

$$\sigma = \begin{matrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{matrix}$$

Les matrices et les vecteurs s'écrivent dans les équations avec les environnements de type `matrix`, le symbole `&` est le séparateur de colonnes et `\\` sépare les lignes (`\tabulnewline` peut être utilisé à la place). Il existe différents environnements permettant de réaliser des choix esthétiques pour la représentation des matrices, qui sont chargés à partir de l'extension `amsmath`, comme `pmatrix`, `bmatrix`.

Testez les variantes suivantes - [↗](#)

<code>\begin{matrix}</code>	<code>\begin{pmatrix}</code>	<code>\begin{bmatrix}</code>	<code>\begin{Vmatrix}</code>
<code>a & b \\</code>	<code>a & b \\</code>	<code>a & b \\</code>	<code>a & b \\</code>
<code>c & d</code>	<code>c & d</code>	<code>c & d</code>	<code>c & d</code>
<code>\end{matrix}</code>	<code>\end{pmatrix}</code>	<code>\end{bmatrix}</code>	<code>\end{Vmatrix}</code>

Les tableaux, à l'instar des images sont des flottants. Si ceci reste flou pour vous, nous vous invitons à relire les notes consacrées aux flottants à la séance précédente.

Les tableaux sont gérés nativement sous L^AT_EX, cependant on se retrouve vite limité du point de vue de la mise en forme, mais les extensions suivantes permettent de corriger en partie ce détail.

Extensions supplémentaires pour gérer les tableaux -

```
\usepackage{multirow} % Fusion de cellules
\usepackage{tabularx} % Calcul automatique de la largeur des colonnes
\usepackage{array} % Options de mise en forme plus élaborées
\usepackage{booktabs} % Améliore le remplissage des cellules et
    permet d'affiner le style des lignes de séparation des rangées
```

Les tableaux s'insèrent en utilisant l'environnement `table`, qui sert à légender et dedans l'environnement `tabular` qui contient les données du tableau en tant que telles. La commande `\label` permet de créer une référence vers le premier `\caption` précédent cette commande.

Exemple de tableau -

```
\begin{table}[h!]
\caption{Effets de l'overdose de croquettes sur des chats gourmands}

\label{tab:XP-chats}
\begin{tabular}{lcc}
\hline
\multirow{2}{*}{Sujet} & \multicolumn{2}{c}{Poids (kg)} \\
& Avant & Après \\
\hline
\hline
Mistigri & 3.58 & 4.42 \\
Garfield & 3.88 & 4.56 \\
Rourou & 3.65 & 4.69 \\
Simba & 4.12 & 4.21 \\
\hline
Tendance &  $\pm 3.81 \pm 0.22$  &  $\pm 4.47 \pm 0.17$  \\
\hline
\end{tabular}
\end{table}
```

Voici un autre exemple de tableau un peu plus simple :

TABLE 1 – Tableau très minimaliste...

1	2	3	4
5	6	7	8

- Essayez de le coder vous-même

- La position du texte dans chaque colonne est l'argument de l'environnement `tabular`. Tester l'ajout d'un séparateur vertical en ajoutant en modifiant l'argument en `l|cc`.
- Les colonnes (resp. les lignes) sont séparées par `&` (resp `//` ou `\tabularnewline`). Pour un report de ligne à l'intérieur d'une cellule, utilisez `\newline`.
- Bordure avec `\hline`. Tester la commande `\cline{2-3}`
- Fusion de cellules via `multirow` et `multicolumn`.

En chargeant le package `multirow`, il est possible de fusionner les cellules d'un tableau. Fusionner plusieurs cellules se réalise au moyen des commandes ci-dessous :

- Fusionner des colonnes `\multicolumn{nombredecolonnes}{positionnement}{contenu}`
- Fusionner des lignes `\multirow{nombredelignes}{*}{contenu}`
- Tracer une ligne de la colonne `i` à la colonne `j` `\cline{i-j}`.

Un exemple illustrant la puissance du concept... -

```
\FloatBarrier
\begin{table}[h!]
  \centering
  \caption{Fusion...}
  \subfloat[] {
    \begin{tabular}{cc|c}      % | Ligne verticale entre les colonnes
      \hline                  % Ligne horizontale
      Vegeta & Goku & & Broly \\
      \hline      \hline
      1 & 2 & 3 & \\      % & : sépare les colonnes
      4 & 5 & 6 & \\      % \\ : sépare les lignes
    \hline
    \end{tabular}}
  \subfloat[] {
    \begin{tabular}{ccc}
      \hline
      \multicolumn{2}{c}{Gogeta} & Broly \\      % Fusion de Vegeta et
      & & Goku
    \hline
    \multicolumn{2}{c}{\multirow{2}{*}{9}} & 3 \\      % Ordre : on
      & & fusionne les colonnes puis on fusionne les lignes
      & & 6 \\
    \hline
    \end{tabular}}
\end{table}
\FloatBarrier
```

Les options (arguments) de mise en forme, `\begin{tabular}{.....}` sont résumées dans la Table 2.

TABLE 2 – Liste des options de mise en forme par colonne

l	Texte aligné à gauche	Natif
r	Texte aligné à droite	
c	Texte centré dans la colonne	
p{2cm}	Fixe la largeur de la colonne à 2cm et place le texte à gauche	array
m{2cm}	Fixe la largeur de la colonne à 2cm et centre le texte verticalement	
b{2cm}	Fixe la largeur de la colonne à 2cm et groupe le texte vers le bas	
X	Calcul automatique de la dimension de la colonne	tabularx

Vous aurez remarqué que toutes les combinaisons d’alignements horizontaux/verticaux ne sont pas couvertes par ces options... Cependant les commandes `\centering`, `raggedright` et `raggedleft` permettent de contrôler l’alignement horizontal du texte lors de l’utilisation de p, m et b. Pour injecter du texte ou une commande automatiquement au début (resp. à la fin) du contenu de chaque cellule d’une colonne de type Z, il est nécessaire d’utiliser les descripteurs de textes : préciser dans les arguments du tableau `>{contenu incroyable}Z` (resp. `Z<{contenu fantastique}`). L’exemple suivant est plus parlant :

Un exemple illustrant la puissance du concept... - [↗](#)

```
\begin{tabular}{|c<{\degres C}|>{l’eau }l<{.}|}|
\hline
-1 & gel \\ \hline
90 & bout \\ \hline
22 & est bonne \\ \hline
\end{tabular}
```

Pour personnaliser le séparateur de colonnes, il est possible de remplacer les `|` par `!Texte servant de séparateur` (ou par `@Texte servant de séparateur`, testez les deux opérateurs pour comprendre la nuance) :

Un exemple illustrant la puissance du concept... - [↗](#)

```
\begin{tabular}{|l!{$\rightarrow$}c!{\ding{37}}l|}|
\hline
Robert Bidochon&RB&03.03.03.03.01\\\hline
Ginette Lacaille&GL&02.02.02.02.01\\\hline
\end{tabular}
```

Pour insérer des lignes de séparation d’épaisseur variable, il est nécessaire d’utiliser le package `booktabs`. L’exemple suivant est assez parlant :

Exemple de séparateur avec booktabs -

```

\begin{table}\centering
\begin{tabular}{lrc}
\toprule[2pt]
a & b & c \\
\cmidrule[1pt](rl){1-3}
d & e & f \\
j & k & l \\
\addlinespace[0.5em]
g & h & i \\
\bottomrule[2pt]
\end{tabular}\caption{abcde}
\end{table}

```

L'utilisation de ces commandes se réalise dans la cellule, ce qui peut être fastidieux et rapidement illisible. Heureusement, l'extension `array` permet de définir de nouveaux types de colonne via la commande `\newcolumntype[nargs]{definition_du_type}...`

Par exemple : `\newcolumntype{0}{>\color{blue} \raggedright}m{5cm}<\degrees C}}`

Liste des options de mise en forme pour les séparateurs

Vous l'aurez compris, mettre en forme un grand nombre de colonnes peut se révéler rapidement fastidieux et source d'erreurs de compilation. Cependant il est possible de gérer les alignements d'un grand nombre de colonnes simultanément grâce à l'utilisation de la syntaxe suivante

`*{nombreColonnes}{formatColonne}`.

- `{|*{5}{p{2cm}}|}}` est équivalent à
`{|p{2cm}|p{2cm}|p{2cm}|p{2cm}|p{2cm}|}`
- `{|*{3}{c}|*{2}{p{4cm}}|1|*{3}{p{2cm}}|}}` est équivalent à
`{|c|c|c|p{4cm}|p{4cm}|1|p{2cm}|p{2cm}|p{2cm}|}`

Vous avez désormais un aperçu global des manipulations nécessaires pour manipuler les tableaux. Les tableaux de grandes tailles restent cependant compliqués à mettre en forme manuellement. Heureusement il existe des outils pour éviter de devoir réaliser ces manipulations en lignes de code.

Générer des tableaux $\text{\LaTeX}$ sans coder (ou presque) !

Vous l'aurez compris, écrire des tableaux en $\text{\LaTeX}$ n'est pas l'aspect le plus fun de la rédaction quand il s'agit d'ajouter des tableaux de grandes tailles.

Heureusement d'autres personnes ont vécu cette frustration avant vous et ont créé des macros de conversion pour passer de tableaux codés en fichiers Excel/Calc en format utilisable par $\text{\LaTeX}$. Ces outils sont disponibles ici :

- <https://ctan.org/tex-archive/support/excel2latex/>
- <https://sourceforge.net/projects/calc2latex/>

Le site <https://www.tablesgenerator.com> permet de réaliser la conversion en copiant collant un tableau Excel. Il est possible d'y fusionner directement des lignes et de personnaliser les séparateurs.

2 Les projets en L^AT_EX

Lorsqu'un document de grande taille est créé, il convient de choisir la classe du document de façon adéquate. Typiquement le choix s'effectue entre **book**, **report** et **memoir**.

Pour rendre le code lisible et facilement manipulable, il est nécessaire de scinder le projet global en plusieurs fichiers `.tex`. Le préambule est généralement isolé du reste du projet, tandis qu'un dossier différent est créé pour chaque partie du projet. Un set de commandes particulières permet de gérer les interactions entre les différentes parties.

Deux packages sont principalement utilisés pour réaliser ces opérations : **subfiles** et **import**. Les deux ne sont pas incompatibles, mais ils ont chacun leurs avantages, ils peuvent cohabiter ensemble, à chaque utilisateur d'en tirer le meilleur.

subfiles permet de compiler des sous-fichiers indépendamment du fichier maître, ce qui est généralement plus ergonomique. **import** permet d'inclure facilement des fichiers contenant du code `.tex` que l'on a pas l'intention de compiler seuls. Ce package ne fonctionne pas uniquement que pour les fichiers `.tex`, mais également pour les `pdf/pdf_tex`, ce qui peut s'avérer très utile.

2.1 Paramétrage du projet et des sous-fichiers

2.1.1 **import**

Pour les gros fichiers, les codes finissent souvent par s'encombrer un peu. Il est alors possible de les partitionner pour travailler sur plusieurs documents. Inclure un fichier dans un autre s'effectue au moyen du package **import**. Deux commandes proches existent :

commande **import**

- `\import{chemin_vers_le_fichier}{nom_du_fichier.tex}`
- `\subimport{chemin_vers_le_fichier}{nom_du_fichier.tex}`

Les deux commandes fonctionnent avec des liens relatifs ou absolus. La différence réside dans le fait que la commande `\subimport` permet d'inclure un sous-fichier contenant des sous-sous-fichiers, tandis qu'il n'est pas possible d'utiliser **import** dans un fichier préalablement importé via **import**. Cela s'appelle l'inclusion imbriquée. Cette commande permet de scinder le travail mais ne divise pas la compilation. Les fichiers sont considérés comme faisant partie du corps du document, il n'est pas nécessaire d'inclure un préambule.

2.1.2 **subfiles**

Cet outil permet de partitionner un fichier sans recompiler à chaque fois l'intégralité du projet et visualiser le travail de rédactions en plus petits documents. Le préambule s'écrit dans un document principal, mais chaque sous-fichier comprend un préambule simplifié où on fait référence au document maître, mais le contenu de chaque fichier est écrit dans un environnement **document**. Un exemple est affiché ci-après.

Utilisation de subfiles

Dans le préambule du document maître :

```
\documentclass{ma_classe_de_document}  
\usepackage{subfiles}
```

Dans le corps du document maître, inclure un fichier .tex dans un autre peut s'effectuer via la commande `\subfile`.

```
\subfile{dossier/sous-dossier/mon_fichier.tex}
```

Dans les sous-fichiers :

```
\documentclass[nom_du_fichier_principal.tex]{subfiles}  
% Ceci permet de faire appel au préambule du fichier main.  
% Et de compiler le sous-fichier indépendamment du reste...  
\begin{document}  
...  
\end{document}
```

Pour que les documents esclaves soient compilables indépendamment, il est nécessaire que la position relative du document maître soit écrite dans leur code. Si ceux-ci se trouvent imbriqués dans un sous-dossier, le document maître est compilable, mais pas les documents esclaves, à moins de l'écrire dans les options du document via le chemin relatif `../! ./` signifie que le document esclave se trouve dans le même dossier que le document maître, chaque point signifie que le fichier esclave descend dans l'arborescence de dossier par rapport au document maître...

Utilisation de subfiles - Dossiers imbriqués

```
\documentclass[../nom_du_fichier_principal.tex]{subfiles}  
% Ceci permet de faire appel au préambule du fichier main.  
% Et de compiler le sous-fichier indépendamment du reste...  
\begin{document}  
...  
\end{document}
```

Annexes

A Inclure un .csv (ou .dat) dans son document sans suer

A découvrir le plus vite possible¹...

Le package `pgfplotstable` permet d'inclure et mettre en forme un fichier sous forme d'un tableau sans options de mise en forme dans son document. Notons la compatibilité du package avec les packages `siunitx` pour automatiser les notations scientifiques et `longtable` pour les tableaux très très longs (introduction en fin d'annexe). Cela demande une prise en main plus longue que les quelques minutes consacrées en séance, mais voici quelques exemples simples mais parlant.

Exemple de tableau court sous `pgfplotstable` -

```
\documentclass{article}
\usepackage{pgfplotstable}
\begin{document}
\pgfplotstabletypeset[
  col sep=comma,
  string type,
  every head row/.style={%
    before row={\hline
      \multicolumn{2}{c}{Full Name} & \\
    },
    after row=\hline
  },
  every last row/.style={after row=\hline},
  columns/name/.style={column name=Name, column type=l},
  columns/surname/.style={column name=Surname, column type=l},
  columns/age/.style={column name=Age, column type=c},
]{scientists.csv}
\end{document}
```

1. Découverte de la soirée d'avant présentation, toutes ces années en passant à côté de cet outil magnifique, ma perception du monde est aujourd'hui bien différent d'hier... 

Exemple d'utilisation de siunitx sous pgfplotstable via le type de colonne S -

```

\documentclass{article}

\usepackage{siunitx} % Formats the units and values
\usepackage{pgfplotstable} % Generates table from .csv
% Setup siunitx:
\sisetup{
  round-mode          = places, % Rounds numbers
  round-precision     = 2, % to 2 places
}

\begin{document}

  \pgfplotstabletypeset[
    multicolumn names,
    col sep=comma, % the separator in our .csv file
    string type, % Added in hopes of enabling alphabetic input.
    display columns/0/.style={
      column name=\textit{Value 1}, % name of first column
      string type, % use siunitx for formatting
    }
    display columns/1/.style={
      column name=\textit{Value 2},
      column type={S[table-format=2.2e2]}
    }
  ]{
Two,2.000000E+015
Three,2.000000E+015
  }

\end{document}

```

Exemple de longtable sous pgfplotstable -

```
\documentclass{article}

\usepackage{pgfplotstable}
\usepackage{longtable}
\usepackage{booktabs}
\usepackage{array}
\usepackage{siunitx}

\begin{document}
  \pgfplotstableset{
    begin table=\begin{longtable},
    end table=\end{longtable},
  }
  \pgfplotstabletypeset[
    columns/temperature/.style={
      string type,
      column name=Temperature(K)
    },
    columns/value/.style={
      string type,
      column name=Value($m^3$)
    },
    every head row/.append style={before row={\caption{Some
      caption}}\label{tab:sometable}\\\toprule},after row=\
      midrule\endfirsthead},
    every first row/.append style={before row={\multicolumn{2}{c}
      }}\ \caption[] {Some caption}\\\toprule},after row=\
      midrule\endhead},
    every last row/.style={after row=\bottomrule},
  ]{sometable.dat}

  Refer to the table here \ref{tab:sometable}.
\end{document}
```

B Comment exporter facilement des tables produites sous R vers le LaTeX ?

L'extension `stargazer` doit-être préalablement installée sous R. Cette extension permet d'exporter une grande diversité d'objets, allant du dataframe au résultat d'une régression, l'option «summary» permet de synthétiser l'information facilement.

<https://www.r-bloggers.com/stargazer-package-for-beautiful-latex-tables-from-r-statistical-models-output/>

C Comment exporter facilement des tables produites sous Matlab vers le LaTeX ?

S'il existe qu'un outil natif pour exporter les matrices en matlab (`latex`), il n'offre pas la même ergonomie que Stargazer sous R, il n'est pas forcément très adapté aux structures et aux tables Matlab. Il existe cependant un outil mis en ligne sur le site d'échange disponible via le lien ci-dessous.

<https://nl.mathworks.com/matlabcentral/fileexchange/69063-matlab-table-to-latex-conversor>

D Comment exporter facilement des tables produites sous Python vers le LaTeX ?

- Pour un objet de type `pandas` : la fonction `tondataframe.to_latex()` fera le travail, elle est déjà implémentée dans la bibliothèque `pandas`. Veillez à inclure `booktabs` dans votre préambule.
- Exporter une matrice sous `numpy` : pas de fonction native, mais la bibliothèque `array-to-latex` disponible sur `pip`.

E Changer l'orientation d'une page, passer du mode portrait en mode paysage sur une partie du document

Il est nécessaire d'insérer dans le préambule l'extension `lscape`. Tout ce qui se trouve dans l'environnement `landscape` est en paysage :

```
\begin{landscape}  
Contenu en orientation paysage  
\end{landscape}
```

F Colorer les lignes d'un tableau

Il faut avant toute chose charger `xcolor` en plaçant à la seconde ligne.

```
\usepackage[table,x11names,dvipsnames,table]{xcolor}
```

Pour colorer une seule ligne, utiliser la commande suivante juste avant la ligne à colorer.

```
\rowcolor{couleur} % couleur : par exemple red!25, rouge avec opacité de  
25%
```

Pour faciliter la lecture d'un tableau de grande taille, il est parfois préconisé de colorer une ligne sur deux d'un tableau en insérant avant le `tabular` ou le `longtable`

```
\rowcolors{indice de début}{couleur indices pairs}{couleur indices  
impairs} % Indice débutant à 1 !
```

G Inclure des tableaux de grande taille dans un code $\text{\LaTeX}$

Il est nécessaire de faire appel à l'extension `longtable` et à l'environnement du même nom. `booktabs` doit également être inclus dans le document.

`longtable` ne doit pas être inclus dans un environnement `table`, on peut directement inclure la commande `\caption` à l'intérieur et pour disposer le `longtable` sur la page, il faut utiliser les environnements utilisés avec du contenu plus classique (`center`, `flushright` et `flushleft`).

Voici quelques commandes utiles :

```
\endfirsthead % fin de ligne apparaissant au sommet de la première page
\endhead % fin de la ligne apparaissant au sommet de chaque page
\endfoot % fin de la ligne apparaissant à la fin de chaque page
\endlastfoot % fin de la ligne apparaissant au pied de la dernière page
```

Exercice : inclure le tableau fourni dans le fichier `hugeTabColinet.tex`. Une fois que vous maîtriserez le concept, répétez l'opération avec `hugeTabColinet.csv` et `pgfplotstable`