L'extension nicematrix *

F. Pantigny
fpantigny@wanadoo.fr

17 décembre 2025

Résumé

L'extension LaTeX nicematrix fournit de nouveaux environnements sim
ments classiques {tabular}, {array} et {matrix} de array et amsmath mai
lités plus étendues.

$$\begin{array}{c} \\ L_1 \\ L_2 \\ \vdots \\ \vdots \\ L_n \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \cdots & C_n \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \end{array}$$

	dimensions (cm)			Produit
	L	l	h	Prix
petit	3	5.5	1	30
standard	5.5	8	1.5	50.5
premium	8.5	10.5	2	80
extra	8.5	10	1.5	85.5
spécial	12	12	0.5	70

L'extension nicematrix est entièrement contenue dans le fichier `nicematrix.sty`. Ce fichier peut être placé dans le répertoire courant ou dans une arborescence `texmf`. Le mieux reste néanmoins d'installer nicematrix avec une distribution TeX comme MiKTeX, TeX Live ou MacTeX.

Remarque : Si vous utilisez un service LaTeX via Internet (ex. : Overleaf ou TeXPage) vous pouvez télécharger le fichier `nicematrix.sty` dans le dossier de votre projet pour bénéficier de la dernière version de nicematrix.[1]

Cette extension peut être utilisée avec `xelatex`, `lualatex` et `pdflatex` mais aussi avec le cheminement classique `latex-dvips-ps2pdf` (ou Adobe Distiller). *Néanmoins, le fichier nicematrix-french.tex de la présente documentation ne peut être compilé qu'avec LuaLaTeX.*

Cette extension nécessite et charge les extensions array, amsmath et pgfcore ainsi que le module shapes de PGF (l'extension tikz, qui est une surcouche de PGF, n'est *pas* chargée). L'utilisateur final n'a qu'à charger l'extension nicematrix avec l'instruction habituelle : `\usepackage{nicematrix}`.

L'idée de nicematrix est de créer des nœuds PGF derrière les cases et les positions des filets des tableaux créés par array et de les utiliser pour développer de nouvelles fonctionnalités. Comme toujours avec PGF, les coordonnées de ces nœuds sont écrites dans le fichier `aux` pour être utilisées à la compilation suivante. C'est pourquoi l'utilisation de nicematrix nécessite **plusieurs compilations successives**[2]. L'utilisateur ne doit pas utiliser la commande `\nofiles` (qui bloque l'écriture du fichier `aux`).

La plupart des fonctionnalités de nicematrix sont accessibles sans avoir à utiliser explicitement PGF ou TikZ (ce dernier n'est d'ailleurs pas chargé par défaut).

Une commande `\NiceMatrixOptions` est fournie pour régler les options (la portée des options fixées par cette commande est le groupe TeX courant : elles sont semi-globales).

*Ce document correspond à la version 7.5 de nicematrix, en date du 2025/12/14.

[1] La dernière version de `nicematrix.sty` peut être téléchargée sur le dépôt Github de nicematrix : https://github.com/fpantigny/nicematrix/releases

[2] Si vous utilisez Overleaf, Overleaf effectue automatiquement un nombre de compilations suffisant (en utilisant `latexmk`).

Table des matières

1 Les environnements de cette extension

L'extension nicematrix définit les nouveaux environnements suivants :

{NiceTabular}	{NiceArray}	{NiceMatrix}	{NiceArrayWithDelims}
{NiceTabular*}	{pNiceArray}	{pNiceMatrix}	
{NiceTabularX}	{bNiceArray}	{bNiceMatrix}	
	{BNiceArray}	{BNiceMatrix}	
	{vNiceArray}	{vNiceMatrix}	
	{VNiceArray}	{VNiceMatrix}	

Les environnements {NiceArray}, {NiceTabular} et {NiceTabular*} sont similaires aux environnements {array}, {tabular} et {tabular*} de l'extension array (qui est chargée par nicematrix).

Les environnements {pNiceArray}, {bNiceArray}, etc. n'ont pas d'équivalents dans array.

Les environnements {NiceMatrix}, {pNiceMatrix}, etc. sont similaires aux environnements correspondants de l'amsmath (qui est chargée par nicematrix) : {matrix}, {pmatrix}, etc.

L'environnement {NiceTabularX} est similaire à l'environnement {tabularx} de l'extension éponyme.[3]

L'environnement {NiceArrayWithDelims} est une généralisation de {NiceArray} et ses variantes (cf. p. 54).

On conseille d'utiliser prioritairement les environnements classiques et de n'utiliser les environnements de nicematrix que lorsqu'on utilise les fonctionnalités supplémentaires offertes par ces environnements (cela permet d'économiser la mémoire et d'accélérer la compilation).[4]

Tous les environnements de l'extension nicematrix acceptent, entre crochets, une liste optionnelle de paires de la forme *clé=valeur*. **Il doit n'y avoir aucun espace devant le crochet ouvrant ([) de cette liste d'options.**

2 L'espace vertical entre les rangées

Il est bien connu que certaines rangées[5] des tableaux créés par défaut avec LaTeX sont trop proches l'une de l'autre. On en donne ci-dessous un exemple classique.

```
$\begin{pmatrix}
\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{3} & \frac{1}{4} \\
\end{pmatrix}$
```

$$\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{3} & \frac{1}{4} \end{pmatrix}$$

En s'inspirant de l'extension cellspace qui traite de ce problème, l'extension nicematrix propose deux clés `cell-space-top-limit` et `cell-space-bottom-limit` qui sont similaires aux deux paramètres \cellspacetoplimit et \cellspacebottomlimit proposés par cellspace.
Il existe aussi une clé `cell-space-limits` pour régler simultanément les deux paramètres.
La valeur initiale de ces paramètres est 0 pt pour que les environnements de nicematrix aient par défaut le même comportement que ceux de array et de l'amsmath mais une valeur de 1 pt serait un bon choix. On conseille de régler leurs valeurs avec la commande \NiceMatrixOptions.[6]

[3] Néanmoins, on peut aussi utiliser directement les colonnes X dans l'environnement {NiceTabular}, la largeur souhaitée pour le tableau étant spécifiée par la clé width : cf. p. 30.

[4] Pour accélérer les compilations, on peut aussi utiliser ponctuellement la clé no-cell-nodes qui supprime la création des nœuds PGF-TikZ correspondants aux contenus des cellules mais certains fonctionnalités ne seront plus disponibles (cf. 55).

[5] Dans ce document, on parlera de *rangée* pour désigner uniquement les rangées horizontales. Les colonnes, par opposition, sont verticales.

[6] On remarquera que ces paramètres s'appliquent aussi aux colonnes de type S de siunitx alors que cellspace n'est pas utilisable avec ces colonnes.

```
\NiceMatrixOptions{cell-space-limits = 1pt}

$\begin{pNiceMatrix}
\frac{1}{2} & -\frac{1}{2} \\
\frac{1}{3} & \frac{1}{4} \\
\end{pNiceMatrix}$
```

$$\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{3} & \frac{1}{4} \end{pmatrix}$$

Il est également possible de changer ces paramètres pour certaines rangées du tableau seulement grâce à la commande `\RowStyle` (cf. p. 27).

3 La clé baseline

L'extension nicematrix propose une option `baseline` pour la position verticale des tableaux. Cette option `baseline` prend comme valeur un entier qui indique le numéro de rangée dont la ligne de base servira de ligne de base pour le tableau.

```
$A = \begin{pNiceMatrix}[baseline=2]
\frac{1}{\sqrt{1+p^2}} & p & 1-p \\
1 & 1 & 1 \\
1 & p & 1+p
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} \frac{1}{\sqrt{1+p^2}} & p & 1-p \\ 1 & 1 & 1 \\ 1 & p & 1+p \end{pmatrix}$$

L'option `baseline` peut aussi prendre les trois valeurs spéciales `t`, `c` et `b`. Ces trois lettres peuvent aussi être utilisées de manière absolue comme pour l'option des environnements `{tabular}` et `{array}` tels que définis par l'extension array. La valeur initiale de `baseline` est `c`.

Dans l'exemple suivant, on utilise l'option `t` (synonyme de `baseline=t`) immédiatement après un `\item` de liste. On remarquera que la présence d'un `\hline` initial n'empêche pas l'alignement sur la ligne de base de la première rangée (avec `{tabular}` ou `{array}` de l'extension array, il faut utiliser `\firsthline`).

```
\begin{enumerate}
\item un item
\smallskip
\item \renewcommand{\arraystretch}{1.2}
$\begin{NiceArray}[t]{lcccccc}
\hline
n   & 0 & 1 & 2 & 3 & 4  & 5 \\
u_n & 1 & 2 & 4 & 8 & 16 & 32
\hline
\end{NiceArray}$
\end{enumerate}
```

1. un item
2.

n	0	1	2	3	4	5
u_n	1	2	4	8	16	32

Il est également possible d'utiliser les outils de booktabs : `\toprule`, `\bottomrule`, `\midrule`, etc., à condition, bien entendu, d'avoir chargé booktabs.

```
\begin{enumerate}
\item an item
\smallskip
\item
$\begin{NiceArray}[t]{lcccccc}
\toprule
n   & 0 & 1 & 2 & 3 & 4  & 5 \\
\midrule
u_n & 1 & 2 & 4 & 8 & 16 & 32
\bottomrule
\end{NiceArray}$
\end{enumerate}
```

1. an item
2.

n	0	1	2	3	4	5
u_n	1	2	4	8	16	32

On peut aussi utiliser la clé `baseline` pour aligner une matrice sur un filet horizontal (tracé par `\hline`). On doit pour cela donner la valeur `line-`i où i est le numéro de la rangée qui *suit* ce filet horizontal.

```
\NiceMatrixOptions{cell-space-limits=1pt}

$A=\begin{pNiceArray}{cc|cc}[baseline=line-3]
\dfrac{1}{A} & \dfrac{1}{B} & 0 & 0 \\
\dfrac{1}{C} & \dfrac{1}{D} & 0 & 0 \\
\hline
0 & 0 & A & B \\
0 & 0 & D & D \\
\end{pNiceArray}$
```

$$A = \left(\begin{array}{cc|cc} \frac{1}{A} & \frac{1}{B} & 0 & 0 \\ \frac{1}{C} & \frac{1}{D} & 0 & 0 \\ \hline 0 & 0 & A & B \\ 0 & 0 & D & D \end{array}\right)$$

4 Les blocs

4.1 Cas général

Dans les environnements de nicematrix, on peut utiliser la commande `\Block` pour placer un élément au centre d'un rectangle de cases fusionnées.[7]
La commande `\Block` doit être utilisée dans la case supérieure gauche du bloc avec deux arguments obligatoires.

- Le premier argument est la taille de ce bloc avec la syntaxe i-j où i est le nombre de rangées et j le nombre de colonnes du bloc.

  Si cet argument est laissé blanc, la valeur par défaut est `1-1`. Si le nombre de rangées n'est pas indiqué, ou bien est égal à `*`, le bloc s'étend jusqu'à la dernière rangée (idem pour les colonnes).

[7] Les espaces situés après une commande `\Block` sont supprimés.

- Le deuxième argument est le contenu du bloc.

  Dans `{NiceTabular}`, `{NiceTabular*}` et `{NiceTabularX}`, le contenu est composé en mode texte tandis que, dans les autres environnements, il est composé en mode mathématique.

  Pour des raisons techniques liées à LaTeX, on ne peut pas débuter le contenu du bloc par une commande `\color` (mais on peut envisager d'utiliser `\textcolor`).[8]

Voici un exemple d'utilisation de la commande `\Block` dans une matrice mathématique.

```
$\begin{bNiceArray}{cw{c}{1cm}c|c}[margin]
\Block{3-3}{A} & & & 0 \\
& & & \Vdots \\
& & & 0 \\
\hline
0 & \Cdots& 0 & 0
\end{bNiceArray}$
```

$$\left[\begin{array}{ccc|c} \multicolumn{3}{c|}{\multirow{3}{*}{A}} & 0 \\ & & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array}\right]$$

On peut souhaiter agrandir la taille du « A » placé dans le bloc de l'exemple précédent. Comme il est composé en mode mathématique, on ne peut pas directement utiliser une commande comme `\large`, `\Large` ou `\LARGE`. C'est pourquoi une option à mettre entre chevrons est proposée par `\Block` pour spécifier du code LaTeX qui sera inséré *avant* le début du mode mathématique.[9]

```
$\begin{bNiceArray}{cw{c}{1cm}c|c}[margin]
\Block{3-3}<\LARGE>{A} & & & 0 \\
& & & \Vdots \\
& & & 0 \\
\hline
0 & \Cdots& 0 & 0
\end{bNiceArray}$
```

$$\left[\begin{array}{ccc|c} \multicolumn{3}{c|}{\multirow{3}{*}{\LARGE A}} & 0 \\ & & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & 0 \end{array}\right]$$

La commande `\Block` accepte en premier argument optionnel (entre crochets) une liste de couples *clé=valeur*.

Les premières clés sont des outils rapides pour contrôler l'apparence du bloc :

- la clé `fill` prend en argument une couleur et remplit le bloc avec cette couleur ;
- la clé `opacity` fixe l'opacité de la couleur de remplissage donnée par `fill` ;[10]
- la clé `draw` prend en argument une couleur et trace le cadre avec cette couleur (la valeur par défaut de cette clé est la couleur courante des filets du tableau) ;
- la clé `color` prend en argument une couleur et l'applique au contenu et trace également le cadre avec cette couleur ;
- les clés `hlines`, `vlines` et `hvlines` tracent les filets correspondants dans le bloc[11] ;
- la clé `line-width` fixe la largeur utilisée pour tracer les filets (n'a d'intérêt que si `draw`, `hvlines`, `hlines` ou `vlines` est utilisée) ;
- la clé `rounded-corners` impose des coins arrondis (pour le cadre dessiné par `draw` et le fond dessiné par `fill`) avec un rayon égal à la valeur de cette clé (la valeur par défaut est 4 pt[12]).

[8] cf. https://tex.stackexchange.com/questions/674788

[9] Cet argument entre chevrons peut aussi être utilisé pour insérer une commande de fonte comme `\bfseries`, ce qui peut être utile dans le cas où la commande `\\` apparaît dans le contenu du bloc. On peut aussi y mettre la commmande `\rotate` fournie par nicematrix (cf. partie 14.5, p. 52).

[10] Attention : cette fonctionnalité génère des instructions de transparence dans le PDF résultant et certains lecteurs de PDF n'acceptent pas la transparence. L'application de l'opacité est faite par `\pgfsetfillopacity`

[11] Néanmoins, les filets ne sont pas tracés dans les sous-blocs du bloc, conformément à l'esprit de nicematrix : les filets ne sont pas tracés dans les blocs, sauf s'ils possèdent la clé `transparent` (cf. section 5 p. 12).

[12] Cette valeur par défaut est la valeur initiale des *rounded corners* de TikZ.

Ces outils ne sont parfois pas suffisants pour contrôler l'apparence du bloc. Les clés suivantes sont plus puissantes, mais plus difficiles d'utilisation. Elles nécessitent également que TikZ soit chargé (par `\usepackage{tikz}`). Par défaut, nicematrix ne charge pas TikZ mais uniquement PGF, qui est une sous-couche de TikZ.

- La clé `borders` permet de ne tracer que certaines des bordures du bloc : cette clé prend comme valeur une liste d'éléments parmi les suivants : `left`, `right`, `top` et `bottom` ; on peut en fait, dans la liste qui est la valeur de la clé `borders` mettre une entrée de la forme `tikz={`*liste*`}` où *liste* est une liste de couples *clé=valeur* de TikZ spécifiant les caractéristiques graphiques des traits qui seront dessinés (pour un exemple, voir p. 69).
- Quand la clé `tikz` est utilisée, le chemin TikZ correspondant au rectangle délimitant le bloc est exécuté avec TikZ[13] en utilisant comme options la valeur de cette clé `tikz` (qui doit donc être une liste de clés TikZ applicables à un chemin de TikZ). Pour des exemples d'utilisation de cette clé `tikz`, voir p. 63.

  En fait, dans la liste des clés fournies à `tikz`, on peut mettre une clé `offset`. Cette clé n'est pas fournie par TikZ mais par nicematrix. Elle réduit le rectangle correspondant au bloc par une marge (horizontalement et verticalement) égale à la valeur (passée à `offset`). C'est ce rectangle réduit qui sera le chemin exécuté par TikZ avec comme options les autres clés passées à la clé `tikz`.

Enfin, il existe quelque clés techniques :

- la clé `name` donne un nom au nœud TikZ rectangulaire correspondant au bloc ; on peut utiliser ce nom avec TikZ dans le `\CodeAfter` (cf. p. 40) ;
- la clé `respect-arraystretch` évite la remise à 1 de `\arraystretch` en début de bloc (qui a lieu par défaut) ;
- Par défaut, les filets ne sont pas tracés dans les blocs (voir à ce sujet la partie sur les filets, section 5 p. 12). Néanmoins, si la clé `transparent` est utilisée, les filets seront tracés.[14]

  Pour un exemple, voir la section 18.1, page 63.

  Attention : cette clé n'implique pas du tout que le contenu du bloc sera transparent.

Il existe aussi des clés de positionnement horizontal et vertical du contenu du bloc qui sont décrites ci-dessous (cf. 4.5 p. 8).

On doit remarquer que, par défaut, les blocs ne créent pas d'espace. Il n'y a exception que pour les blocs mono-rangée et les blocs mono-colonne dans certaines conditions comme expliqué plus loin.

Dans l'exemple suivant, on a dû élargir à la main les colonnes 2 et 3 (avec la construction classique `w{c}{...}` de l'extension array).

```
\begin{NiceTabular}{cw{c}{2cm}w{c}{3cm}c}
rose      & tulipe & marguerite & dahlia \\
violette
& \Block[draw=red,fill=[RGB]{204,204,255},rounded-corners]{2-2}
        {\LARGE De très jolies fleurs}
   & & souci \\
pervenche & & & lys \\
arum      & iris & jacinthe & muguet
\end{NiceTabular}
```

rose	tulipe	marguerite	dahlia
violette			souci
pervenche			lys
arum	iris	jacinthe	muguet

[13] TikZ doit être chargé préalablement (par défaut, nicematrix ne charge que PGF), faute de quoi, une erreur sera levée.
[14] Par ailleurs, la commande `\TikzEveryCell` disponible dans le `\CodeAfter` et le `\CodeBefore`, ne s'applique aux blocs avec la clé `transparent`.

4.2 Les blocs mono-colonne

Les blocs mono-colonne ont un comportement spécial.

- La largeur naturelle du contenu de ces blocs est prise en compte pour la largeur de la colonne courante.

  Dans les colonnes à largeur fixée (`p{...}`, `b{...}`, `m{...}`, `w{...}{...}`, `W{...}{...}`, `V{...}`, similaires aux colonnes `V` de varwidth, et `X`, similaires aux colonnes `X` de tabularx), le contenu du bloc est mis en forme comme un paragraphe de cette largeur.

- La spécification d'alignement horizontal donnée par le type de colonne (`c`, `r` ou `l`) est prise en compte pour le bloc. Pour un bloc dans une colonne de type `p{...}`, `b{...}`, `m{...}`, `V{...}` ou `X`, c'est un alignement `c` qui est retenu par défaut. Néanmoins, ces types de colonnes peuvent avoir une option d'alignement (par ex. `p[l]{...}`), et dans ce cas-là, c'est cette option d'alignement qui est transmise au bloc.

  Notons enfin que le bloc peut avoir sa propre spécification d'alignement horizontal : cf. 4.5 p. 8.

- Les spécifications de fontes imposées à une colonne via la construction `>{...}` dans le préambule du tableau sont prises en compte pour les blocs mono-colonne de cette colonne (ce comportement est assez naturel).

```
\begin{NiceTabular}{@{}>{\color{blue}}lr@{}} \hline
\Block{2-1}{Pierre}     & 12 \\
                        & 13 \\ \hline
Jacques                 &  8 \\ \hline
\Block{3-1}{Stéphanie} & 18 \\
                        & 17 \\
                        & 15 \\ \hline
Amélie                  & 20 \\ \hline
Henri                   & 14 \\ \hline
\Block{2-1}{Estelle}    & 15 \\
                        & 19 \\ \hline
\end{NiceTabular}
```

	12
	13
Jacques	8
	18
	17
	15
Amélie	20
Henri	14
	15
	19

4.3 Les blocs mono-rangée

Pour les blocs mono-rangée, la hauteur (*height*) et la profondeur (*depth*) naturelles sont prises en compte pour la hauteur et la largeur de la rangée en cours (comme le fait la commande standard `\multicolumn` de LaTeX), sauf lorsqu'une option de placement vertical a été utilisée pour le bloc (une des clés `t`, `b`, `m`, `T` et `B` décrites à la partie 4.6, p. 10).

4.4 Les blocs mono-case

Les blocs mono-case héritent des caractéristiques des blocs mono-colonne et des blocs mono-rangée.

On pourrait penser que des blocs d'une seule case n'ont aucune utilité mais, en fait, il y a plusieurs situations où leur utilisation peut présenter des avantages.

- Un bloc mono-case permet d'utiliser la commande `\\` pour composer le bloc sur plusieurs lignes de texte.
- On peut couper la case en plusieurs parties avec `&` quand la clé `&-in-blocks` est activée (voir p. 11).
- On peut utiliser l'option d'alignement horizontal du bloc pour déroger à la consigne générale donnée dans le préambule pour cette colonne (cf. 4.5 p. 8).

- On peut tracer un cadre autour du bloc avec la clé `draw` de la commande `\Block` ou colorier le fond avec des bords arrondis avec les clés `fill` et `rounded-corners`.[15]
- On peut tracer une ou plusieurs bordures de la case avec la clé `borders`.

4.5 Positionnement horizontal du contenu des blocs

La commande `\Block` admet les clés `l`, `c` et `r` pour la position horizontale du contenu du bloc (calé à gauche, centré ou bien calé à droite).

```
$\begin{bNiceArray}{cw{c}{1cm}c|c}[margin]
\Block[r]{3-3}<\LARGE>{A} & & & 0 \\
& & & \Vdots \\
& & & 0 \\
\hline
0 & \Cdots& 0 & 0
\end{bNiceArray}$
```

$$\left[\begin{array}{ccc|c} \multicolumn{3}{r|}{\multirow{3}{*}{\LARGE A}} & 0 \\ & & & \Vdots \\ & & & 0 \\ \hline 0 & \Cdots & 0 & 0 \end{array}\right]$$

Par défaut, le positionnement horizontal des contenus des blocs est calculé sur le *contenu* des colonnes impliquées. De ce fait, dans l'exemple suivant, l'en-tête « Premier groupe » est correctement centré même si un espacement des colonnes a été demandé par une instruction comme `!{\qquad}` dans le préambule (ce n'est pas le cas avec `\multicolumn`).

```
\begin{NiceTabular}{@{}c!{\qquad}ccc!{\qquad}ccc@{}}
\toprule
Rang & \Block{1-3}{Premier groupe} & & & \Block{1-3}{Deuxième groupe} \\
     & 1A & 1B & 1C & 2A & 2B & 2C \\
\midrule
 1 & 0.657 & 0.913 & 0.733 & 0.830 & 0.387 & 0.893\\
 2 & 0.343 & 0.537 & 0.655 & 0.690 & 0.471 & 0.333\\
 3 & 0.783 & 0.885 & 0.015 & 0.306 & 0.643 & 0.263\\
 4 & 0.161 & 0.708 & 0.386 & 0.257 & 0.074 & 0.336\\
\bottomrule
\end{NiceTabular}
```

Rang	Premier groupe			Deuxième groupe		
	1A	1B	1C	2A	2B	2C
1	0.657	0.913	0.733	0.830	0.387	0.893
2	0.343	0.537	0.655	0.690	0.471	0.333
3	0.783	0.885	0.015	0.306	0.643	0.263
4	0.161	0.708	0.386	0.257	0.074	0.336

Pour avoir un positionnement horizontal du contenu du bloc qui s'appuie sur les limites des colonnes du tableau LaTeX (et non sur le contenu de ces colonnes), il faut utiliser les clés `L`, `R` et `C` de la commande `\Block`.[16]

Voici le même exemple avec la clé `C` pour le premier bloc.

[15] Pour colorier simplement le fond d'une case, il n'y a pas besoin d'utiliser un bloc mono-case : on peut utiliser la commande `\cellcolor`.

[16] On remarquera que les clés `L`, `R` et `C` nécessitent moins de calculs que les clés `l`, `r` et `c`. Si on tient à l'efficacité, on devrait écrire `\Block[C]` systématiquement par défaut.

```
\begin{NiceTabular}{@{}c!{\qquad}ccc!{\qquad}ccc@{}}
\toprule
Rang & \Block[C]{1-3}{Premier groupe} & & & \Block{1-3}{Deuxième groupe} \\
     & 1A & 1B & 1C & 2A & 2B & 2C \\
\midrule
 1 & 0.657 & 0.913 & 0.733 & 0.830 & 0.387 & 0.893\\
 2 & 0.343 & 0.537 & 0.655 & 0.690 & 0.471 & 0.333\\
 3 & 0.783 & 0.885 & 0.015 & 0.306 & 0.643 & 0.263\\
 4 & 0.161 & 0.708 & 0.386 & 0.257 & 0.074 & 0.336\\
\bottomrule
\end{NiceTabular}
```

Rang						
	1A	1B	1C	2A	2B	2C
1	0.657	0.913	0.733	0.830	0.387	0.893
2	0.343	0.537	0.655	0.690	0.471	0.333
3	0.783	0.885	0.015	0.306	0.643	0.263
4	0.161	0.708	0.386	0.257	0.074	0.336

La commande `\Block` accepte aussi les clés `p` et `j`. Avec la clé `p`, le contenu du bloc est composé comme un paragraphe (de manière similaire à une colonne standard de type `p`). Cette clé peut s'utiliser en conjonction avec les clés `l`, `c` ou `r` et, alors, le paragraphe est composé avec `\raggedright`, `\centering` ou `\raggedleft` (en fait, quand `ragged2e` est chargée, ce sont les commandes `\RaggedRight`, `\Centering` et `\RaggedLeft` fournies par cette extension qui seront utilisées au lieu de `\raggedright`, `\centering` et `\raggedleft`). Avec la clé `j` (qui force la clé `p`), le paragraphe est composé de manière justifiée.

On peut mettre un environnement `{itemize}` ou `{enumerate}` dans un bloc qui utilise la clé `p` ou la clé `j` (dans les autres cas, on aura une erreur : `Not allowed in LR mode`). Dans l'exemple suivant, on a chargé l'extension `enumitem` (pour pouvoir utiliser la clé `left` de l'environnement `{itemize}`).

```
\begin{NiceTabular}[hvlines]{ccc}
un & deux deux & trois trois \\
un &
\Block[p,l]{*-2}{%
\begin{itemize}[left=0pt]
\item un deux trois quatre cinq
\item deux
\item trois
\end{itemize}%
} \\
un & \\
un & \\
un & \\
un & \\
un & \\
un & \\
\end{NiceTabular}
```

un	deux deux	trois trois
un		

- un deux trois quatre cinq
- un deux
- trois

4.6 Positionnement vertical du contenu des blocs

Concernant le positionnement vertical, la commande `\Block` admet les clés `m`, `t`, `b`, `T` et `B`.

- Avec la clé `m`[17], le contenu du bloc est centré verticalement.
- Avec la clé `t`, la ligne de base du contenu du bloc est alignée avec la ligne de base de la première rangée concernée par le bloc.
- Avec la clé `b`, la ligne de base de la dernière rangée du contenu du bloc (rappelons que le contenu du bloc peut comporter plusieurs lignes de texte séparées par `\\`) est alignée avec la ligne de base de la dernière des rangées du tableau impliquées dans le bloc.
- Avec la clé `T`, le contenu du bloc est calé vers le haut.

  Il n'y a pas de marge verticale. Néanmoins, le contenu du bloc est (toujours) composé en interne dans une `{minipage}`, un `{tabular}` ou un `{array}`, ce qui fait qu'il y a souvent déjà une marge. Si besoin est, on peut toujours ajouter un `\strut`.
- Avec la clé `B`, le contenu du bloc est calé vers le bas.

Quand aucune clé n'est donnée, c'est la clé `m` qui s'applique (sauf pour les blocs mono-rangée).

```
\NiceMatrixOptions{rules/color=[gray]{0.75}, hvlines}

\begin{NiceTabular}{ccc}
\Block[fill=red!10,t,l]{4-2}{deux\\lignes}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

Un
deux
trois
quatre
text text

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,b,r]{4-2}{deux\\lignes}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

Un
deux lignes
deux
trois
quatre
text text

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,T,L]{4-2}{deux\\lignes}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

Un
deux
trois
quatre
text text

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,B,R]{4-2}{deux\\lignes}
 & & \Huge Un\\
 & & deux  \\
 & & trois  \\
 & & \Huge quatre  \\
text & text \\
\end{NiceTabular}
```

Un
deux
trois
quatre
text text

[17] Cette clé a un alias : `v-center`.

4.7 \\ et & dans les blocs

L'extension nicematrix offre la possibilité d'utiliser directement \\ et & dans le contenu d'un bloc (dans le but de formater son contenu) mais il y a quelques restrictions.

- On ne doit pas utiliser à la fois & et \\ dans le même bloc.
- Pour \\, il n'y a pas d'autres restrictions. On peut utiliser \\ dans un bloc pour composer du texte sur plusieurs lignes.
- Pour pouvoir utiliser &, la clé `ampersand-in-blocks` (alias : `&-in-blocks`) doit avoir été activée[18]. Le bloc est alors divisé en sous-blocs comme illustré ci-dessous. Attention toutefois : quand `ampersand-in-blocks` est utilisée, l'argument (principal) de la commande `\Block` est découpé syntaxiquement au niveau des esperluettes &, celles entre accolades sont masquées mais pas celles dans un environnement.[19]

L'esperluette & permet de diviser horizontalement un bloc en sous-blocs *de même taille*.

```
\begin{NiceTabular}{ll}%
   [hvlines,ampersand-in-blocks]
  & les cinq premiers entiers naturels \\
3 & \Block{}{un&deux&trois} \\
4 & \Block{}{un&deux&trois&quatre} \\
5 & \Block{}{un&deux&trois&quatre&cinq} \\
\end{NiceTabular}
```

	les cinq premiers entiers naturels
3	un deux trois
4	un deux trois quatre
5	un deux trois quatre cinq

Comme on le voit, chaque bloc (ici, à chaque fois mono-case) a été divisé en sous-cases de *même taille*. Dans le cas présent, on aurait peut-être préféré le codage suivant :

```
\begin{NiceTabular}{lccccc}%
   [hvlines,ampersand-in-blocks]
  & \Block{1-5}{les cinq premiers
                 entiers naturels} \\
3 & \Block{1-5}{un & deux & trois} \\
4 & \Block{1-5}{un&deux&trois&quatre} \\
5 & un & deux & trois
  & \cellcolor{red!15} quatre & cinq \\
\end{NiceTabular}
```

	les cinq premiers entiers naturels				
3	un	deux	trois		
4	un	deux	trois	quatre	
5	un	deux	trois	quatre	cinq

Dans ce codage, il s'agit de blocs de taille `1-5` qui sont coupés en trois et quatre sous-blocs.

On a illustré la possibilité de colorier une sous-cellule en utilisant la commande `\cellcolor`.

5 Les filets horizontaux et verticaux

Les techniques habituelles pour tracer des filets peuvent être utilisées dans les environnements de nicematrix, à l'exception de `\vline`. Il y a néanmoins quelques petites différences de comportement avec les environnements classiques.

[18] Si ce n'est pas le cas, l'utilisation de & dans l'argument principal de la commande `\Block` provoquera une erreur : `! Extra alignment tab has been changed to \cr` de TeX.

[19] On ne peut donc pas écrire : `\Block[ampersand-in-blocks]{}{\begin{array}{cc}1&2\end{array}}`. Bien sûr, on peut le faire sans la clé `ampersand-in-blocks`.

5.1 Quelques différences avec les environnements classiques

5.1.1 Les filets verticaux

Dans les environnements de nicematrix, les filets verticaux spécifiés par | dans le préambule des environnements ne sont jamais coupés, même en cas de rangée incomplète ou de double filet horizontal spécifié par `\hline\hline` (il n'y a pas besoin d'utiliser l'extension hhline).

```
\begin{NiceTabular}{|c|c|} \hline
Premier & Deuxième \\ \hline\hline
Paul \\ \hline
Marie & Pauline \\ \hline
\end{NiceTabular}
```

Premier	Deuxième
Paul	
Marie	Pauline

En revanche, les filets verticaux ne sont pas tracés à l'intérieur des blocs (créés par `\Block` : cf. p. 5) ni dans les coins (dont la création est demandée par la clé `corners` : cf. p. 15), ni dans les éventuelles rangées extérieures (créées par les clés `first-row` et `last-row` : cf. p. 30).

Si vous utilisez booktabs (qui fournit `\toprule`, `\midrule`, `\bottomrule`, etc.) et que vous tenez absolument à mettre des filets verticaux (ce qui est contraire à l'esprit à booktabs), vous constaterez que les filets tracés par nicematrix sont compatibles avec booktabs. Remarquez que nicematrix ne charge *pas* booktabs.

```
$\begin{NiceArray}{c|ccc} \toprule
a & b & c & d \\ \midrule
1 & 2 & 3 & 4 \\
1 & 2 & 3 & 4 \\ \bottomrule
\end{NiceArray}$
```

a	b	c	d
1	2	3	4
1	2	3	4

Il reste néanmoins possible de définir un spécificateur, nommé par exemple `I`, pour tracer des filets verticaux avec le comportement standard de array :

```
\newcolumntype{I}{!{\vrule}}
```

5.1.2 La commande `\cline`

Les traits verticaux et horizontaux que l'on insère avec `\hline` et le spécificateur de colonne « | » de array rendent le tableau plus large ou plus long d'une quantité égale à la largeur du trait (avec array et aussi avec nicematrix).

Pour des raisons historiques, il n'en est pas de même pour la commande `\cline`, comme on peut le voir avec l'exemple suivant.

```
\setlength{\arrayrulewidth}{2pt}
\begin{tabular}{cccc} \hline
A&B&C&D \\ \cline{2-2}
A&B&C&D \\ \hline
\end{tabular}
```

A	B	C	D
A	B	C	D

Dans les environnements de nicematrix, cette situation est corrigée (il est néanmoins possible de revenir au comportement par défaut de `\cline` avec la clé `standard-cline`).

```
\setlength{\arrayrulewidth}{2pt}
\begin{NiceTabular}{cccc} \hline
A&B&C&D \\ \cline{2}
A&B&C&D \\ \hline
\end{NiceTabular}
```

A	B	C	D
A	B	C	D

Dans les environnements de nicematrix, une instruction `\cline{`*i*`}` est équivalente à `\cline{`*i*`-`*i*`}`.

5.2 L'épaisseur et la couleur des filets

Les environnements de nicematrix proposent une clé `rules/width` pour fixer la largeur (on devrait plutôt dire l'épaisseur) des filets dans l'environnement. En fait, cette clé ne fait que fixer la valeur du paramètre dimensionnel de LaTeX `\arrayrulewidth`.

On sait que colortbl propose la commande `\arrayrulecolor` pour spécifier la couleur de ces filets.

Avec nicematrix, il est possible de spécifier une couleur même si colortbl n'est pas chargé. Par souci de compatibilité, la commande est nommée également `\arrayrulecolor`. Néanmoins, nicematrix propose aussi une clé `rules/color`, disponible dans `\NiceMatrixOptions` ou dans un environnement individuel, pour fixer la couleur des filets. Cette clé fixe localement la couleur des filets (alors que la commande `\arrayrulecolor` agit globalement !). Elle est à privilégier.

```
\begin{NiceTabular}{|ccc|}[rules/color=[gray]{0.9},rules/width=1pt]
\hline
rose & tulipe & lys \\
arum & iris & violette \\
muguet & dahlia & souci \\
\hline
\end{NiceTabular}
```

rose	tulipe	lys
arum	iris	violette
muguet	dahlia	souci

En fait, dans cet exemple, au lieu de `\hline`, il aurait mieux valu utiliser la commande `\Hline`, fournie par nicematrix et décrite ci-dessous, car elle garantit un meilleur résultat dans les lecteurs de PDF aux bas niveaux de zoom.

5.3 Les outils de nicematrix pour tracer des filets

Les outils proposés par nicematrix pour tracer des filets sont les suivants :

- les clés `hlines`, `vlines`, `hvlines` et `hvlines-except-borders` ;
- le spécificateur « | » dans le préambule (pour les environnements à préambule) ;
- la commande `\Hline`.

Ces outils ont en commun de ne pas tracer les filets dans les blocs ni dans les coins vides (quand la clé `corners` est utilisée), ni dans les rangées et colonnes extérieures.

- Les blocs en question sont :
  - ceux créés par la commande `\Block`[20] de nicematrix présentée p. 5 ;
  - ceux délimités implicitement par des lignes en pointillés continues, créées par `\Cdots`, `\Vdots`, etc. : cf. p. 32.
- Les coins sont créés par la clé `corners` détaillée un peu plus loin : cf. p. 15.
- Pour les rangées et colonnes extérieures, cf. p. 30.

En particulier, cette remarque montre déjà une différence entre la commande standard `\hline` et la commande `\Hline` proposée par nicematrix.

Par ailleurs, la commande `\Hline` admet entre crochets un argument optionnel qui est une liste de couples *clé=valeur* qui décrivent un filet. Pour la description de ces clés, voir `custom-line`, p. 17.[21]

De même que la commande `\Hline`, le spécificateur « | » admet entre crochets des options qui caractérisent le filet à tracer.

```
\begin{NiceTabular}{| c | [color=blue] c |}
\Hline
a & b \\
\Hline[color=red]
c & d \\
\Hline
\end{NiceTabular}
```

a	b
c	d

[20] Et aussi la commande `\multicolumn` même s'il est recommandé d'utiliser plutôt `\Block` quand on utilise l'extension nicematrix.

[21] Remarque technique. Si l'utilisateur définit une commande par-dessus la commande `\Hline`, il doit veiller à ce qu'elle soit *développable* au sens de TeX (en utilisant `\NewExpandableDocumentCommand` de LaTeX3, `\newcommand` de LaTeX ou même `\def` de TeX). Exemple : `\NewExpandableDocumentCommand{\RedLine}{}{\Hline[color=red]}`

5.3.1 Les clés hlines et vlines

Les clés `hlines` et `vlines` tracent des filets horizontaux et verticaux.
Si aucune valeur n'est donnée, tous les filets sont tracés.

Quand une valeur est présente, il s'agit d'une liste de numéros de filets à tracer.

- Il est possible de mettre des intervalles de numéros de la forme i-j.
- **Nouveau 7.4** : il est possible de mettre des numéros négatifs qui sont alors comptés à partir de la fin.

En fait, pour les environnements avec délimiteurs (comme `{pNiceMatrix}` ou `{bNiceArray}`), la clé `vlines` ne trace pas les filets extérieurs (ce qui est le comportement certainement attendu).

```
$\begin{pNiceMatrix}[vlines,rules/width=0.2pt]
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 2 & 3 & 4 & 5 & 6
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}$$

Même quand la clé `hlines` est utilisée, il reste possible d'utiliser `\Hline\Hline` pour placer un filet double horizontal. De même, on peut mettre `||` dans le préambule (d'un environnement à préambule) pour placer un double filet vertical, même quand la clé `vlines` est utilisée.

```
$\begin{NiceArray}{c||ccccc}[hlines,vlines]
  & a & b & c & d & e \\ \Hline\Hline
x & 0 & 0 & 0 & 0 & 0 \\
y & 0 & 0 & 0 & 0 & 0 \\
z & 0 & 0 & 0 & 0 & 0 \\
\end{NiceArray}$
```

$$\begin{array}{c||ccccc} & a & b & c & d & e \\ \hline\hline x & 0 & 0 & 0 & 0 & 0 \\ y & 0 & 0 & 0 & 0 & 0 \\ z & 0 & 0 & 0 & 0 & 0 \end{array}$$

5.3.2 Les clés hvlines et hvlines-except-borders

La clé `hvlines`, qui ne prend pas de valeur, est la conjonction des clés `hlines` et `vlines`.

```
\begin{NiceTabular}{cccc}[hvlines,rules/color=blue,rules/width=1pt]
rose      & tulipe & marguerite & dahlia \\
violette  & \Block[draw=red]{2-2}{\LARGE fleurs}  & & souci \\
pervenche & & & lys \\
arum      & iris & jacinthe & muguet
\end{NiceTabular}
```

rose	tulipe	marguerite	dahlia
violette			souci
pervenche		fleurs	lys
arum	iris	jacinthe	muguet

On remarquera que quand la clé `rounded-corners` est utilisée pour l'environnement `{NiceTabular}`, la clé `hvlines` trace des coins arrondis pour le tableau : cf. partie 14.1, p. 50.

La clé `hvlines-except-borders` est similaire à la clé `hvlines` mais ne trace pas les filets sur les bords horizontaux et verticaux du tableau. Pour un exemple d'utilisation de cette clé, voir la partie « Exemple d'utilisation avec tcolorbox » p. 64.

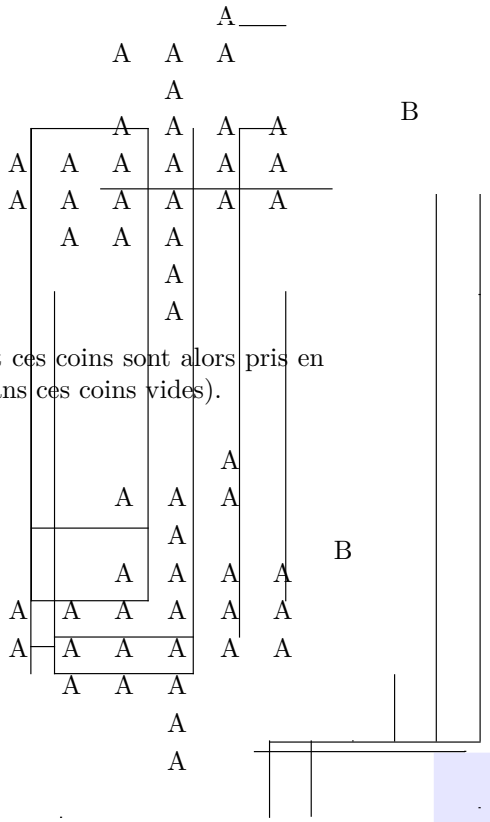
5.3.3 Les coins (vides)

Les quatre coins d'un tableau seront notés `NW`, `SW`, `NE` et `SE` (*north west*, *south west*, *north east* et *south east* en anglais).

Pour chacun de ces coins, on appellera *coin vide* (ou tout simplement *coin*) la réunion de toutes les cases vides situées dans au moins un rectangle entièrement constitué de cases vides partant de ce coin.[22]

On peut néanmoins imposer à une case sans contenu d'être considérée comme non vide par nicematrix avec la commande `\NotEmpty`.

Dans l'exemple ci-contre (où B est au centre d'un `\Block` de taille 2×2), on a colorié en bleu clair les quatre coins (vides) du tableau.



Quand la clé `corners`[23] est utilisée, nicematrix calcule les coins vides et ces coins sont alors pris en compte par les outils de tracés de filets (les filets ne seront pas tracés dans ces coins vides).

```
\NiceMatrixOptions{cell-space-top-limit=3pt}
\begin{NiceTabular}{*{6}{c}}[corners,hvlines]
  &   &   &   & A \\
  &   & A & A & A \\
  &   &   & A \\
  &   & A & A & A & A \\
A & A & A & A & A & A \\
A & A & A & A & A & A \\
  & A & A & A \\
  & \Block{2-2}{B} & & A \\
  &   &   & A \\
\end{NiceTabular}
```

On peut aussi donner comme valeur à la clé `corners` une liste de coins à prendre en considération (les coins sont notés `NW`, `SW`, `NE` et `SE` et doivent être séparés par des virgules).

```
\NiceMatrixOptions{cell-space-top-limit=3pt}
\begin{NiceTabular}{*{6}{c}}[corners=NE,hvlines]
1\\
1&1\\
1&2&1\\
1&3&3&1\\
1&4&6&4&1\\
 & & & & &1
\end{NiceTabular}
```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1

▷ Les coins sont également pris en compte par les outils de coloriage dans le `\CodeBefore`. Ces outils ne colorient pas les cases qui sont dans les coins (cf. p. 20). La commande `\TikzEveryCell` disponible dans le `\CodeAfter` et le `\CodeBefore` (cf. p. 45) tient également compte des coins.

5.3.4 La commande `\diagbox`

La commande `\diagbox` (inspirée par l'extension diagbox) permet, quand elle est utilisée dans une case, de couper cette case selon une diagonale descendante.

[22] Pour être complet, on doit préciser que toute case située dans un bloc (même si elle est vide) n'est pas prise en compte pour la détermination des coins. Ce comportement est naturel. La définition précise de ce qui est considéré comme une « case vide » est donnée plus loin (cf. p. 61).

[23] La clé `corners` dont on parle là n'a pas de rapport direct avec la clé `rounded-corners`, décrite dans la partie 14.1, p. 50.

```
$\begin{NiceArray}{*{5}{c}}[hvlines]
\diagbox{x}{y} & e & a & b & c \\
e & e & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
```

	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Cette commande `\diagbox` peut aussi être utilisée dans un `\Block`.

```
$\begin{NiceArray}{*{5}{c}}[hvlines]
\Block{2-2}{\diagbox{x}{y}} &  & a & b & c \\
  &  & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
```

		a	b	c
		a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Mais on peut aussi utiliser `\diagbox` uniquement pour le trait diagonal et placer les labels comme habituellement.

```
$\begin{NiceArray}{*{5}{c}}[hvlines]
\Block{2-2}{\diagbox{}{}} & y & a & b & c \\
x &  & a & b & c \\
a & a & e & c & b \\
b & b & c & e & a \\
c & c & b & a & e
\end{NiceArray}$
```

		y	a	b	c
x			a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Il est de toutes manières toujours possible de tracer tous les traits souhaités avec TikZ dans le `\CodeAfter` (ou le `\CodeBefore`) en utilisant les nœuds PGF-TikZ créés par nicematrix : cf. p. 55.

5.3.5 Commandes pour filets personnalisés

Il est en fait possible de définir des commandes et des lettres pour des filets personnalisés avec la clé `custom-line`, utilisable dans `\NiceMatrixOptions` ou bien dans un environnement. Cette clé prend en argument une liste de paires de la forme *clé=valeur*. Il y a d'abord trois clés pour spécifier les outils qui permettront d'utiliser ce nouveau type de filet.

- la clé `command` indique le nom (*avec* ou *sans* la contre-oblique) d'une commande qui sera créée par nicematrix et que l'utilisateur pourra utiliser pour tracer des filets horizontaux (de manière similaire à `\hline`) ;
- la clé `ccommand` indique le nom (*avec* ou *sans* la contre-oblique) d'une commande qui sera créée par nicematrix et que l'utilisateur pourra utiliser pour tracer des filets horizontaux partiels (de manière similaire à `\cline`, d'où le nom `ccommand`) : l'argument de cette commande sera une liste d'intervalles de colonnes spécifiés par la syntaxe i ou i-j ;[24]
- la clé `letter` prend en argument une lettre[25] qui pourra être utilisée par l'utilisateur dans le préambule d'un environnement à préambule (comme `{NiceTabular}`) pour spécifier un filet vertical.

On traite maintenant de la description du filet elle-même. Les options qui suivent peuvent aussi s'utiliser dans l'argument optionnel d'une commande `\Hline` individuelle ou dans l'argument optionnel d'un spécificateur « | » dans un préambule d'environnement.

Il y a trois possibilités.

[24] Il est recommandé de n'utiliser ces commandes qu'une seule fois par rangée car chaque utilisation crée un espace vertical entre les rangées correspondant à la largeur totale du trait qui sera tracé. De toutes manières, on peut tracer plusieurs filets avec une unique utilisation de la commande.

[25] Les lettres suivantes ne sont pas autorisées : `lcrpmbVX|()[]!@<>`

- *Première possibilité*

  Il est possible de spécifier des filets multiples, colorés avec une couleur entre les filets (comme on peut le faire avec colortbl par exemple).

  – la clé `multiplicity` indique le nombre de traits successifs qui seront tracés : par exemple, une valeur de 2 va créer des filets doubles comme créés en standard par `\hline\hline` ou bien `||` dans le préambule d'un environnement ;
  – la clé `color` fixe la couleur des filets ;
  – la clé `sep-color` fixe la couleur entre deux filets consécutifs (n'a d'intérêt que dans le cas où la clé `multiplicity` est utilisée). Le nom de cette clé est inspirée par la commande `\doublerulesepcolor` de colortbl.

  Ce système permet en particulier de définir des commandes pour tracer des filets avec une couleur spécifique (et ces filets respecteront les blocs et les coins comme les autres filets de nicematrix).

```
\begin{NiceTabular}{lcIcIc}[custom-line = {letter=I, color=blue}]
\hline
          & \Block{1-3}{dimensions} \\
          & L & l & h \\
\hline
Produit A & 3 & 1 & 2 \\
Produit B & 1 & 3 & 4 \\
Produit C & 5 & 4 & 1 \\
\hline
\end{NiceTabular}
```

	L	l	H
Produit A	3	1	2
Produit B	1	3	4
Produit C	5	4	1

  La clé `sep-color` avec la valeur `white` peut être en particulier utile en cas de filet double au-dessus d'une case colorée (pour éviter que la couleur ne s'applique aussi entre les deux filets).

```
\NiceMatrixOptions
  {
    custom-line =
     {
       command = \DoubleRule ,
       multiplicity = 2 ,
       sep-color = white
     }
  }

\begin{NiceTabular}{ccc}
un & deux & trois \\
\DoubleRule
quatre & \cellcolor{yellow} cinq & six \\
\end{NiceTabular}
```

~~un~~	~~deux~~	trois
quatre	cinq	six

- *Deuxième possibilité*

  On peut utiliser la clé `tikz` (si TikZ est chargé, nicematrix ne chargeant par défaut que PGF). Dans ce cas-là, le filet est tracé directement avec TikZ en utilisant comme paramètres la valeur de la clé `tikz` qui doit être une liste de couples *clé=valeur* applicables à un chemin TikZ.

Par défaut, aucune réservation de place n'est faite pour le filet qui sera tracé avec TikZ. On peut demander une réservation (horizontale pour un filet vertical et verticale pour un filet horizontal) avec la clé `total-width` qui est donc en quelque sorte la largeur du filet qui sera tracé (cette largeur n'est *pas* calculée à partir des caractéristiques fournies par la clé `tikz`).

Voici ce que l'on obtient avec la clé `dotted` de TikZ.

```
\NiceMatrixOptions
  {
    custom-line =
     {
       letter = I ,
       tikz = dotted ,
       total-width = \pgflinewidth
     }
  }

\begin{NiceTabular}{cIcIc}
un & deux & trois \\
quatre & cinq & six \\
sept & huit & neuf
\end{NiceTabular}
```

un	deux	trois
quatre	cinq	six
sept	huit	neuf

- *Troisième possibilité* : la clé `dotted`

  Comme on le voit dans l'exemple précédent, les pointillés tracés par la clé `dotted` de TikZ ne sont pas ronds. C'est pourquoi l'extension nicematrix propose dans la clé `custom-line` une clé `dotted` qui va tracer des pointillés ronds. La valeur initiale de la clé `total-width` est, dans ce cas-là, égale au diamètre des points (l'utilisateur peut quand même utiliser la clé `total-width` pour en changer la valeur). Ces pointillés ronds sont aussi utilisés par nicematrix pour des lignes en pointillés continues créées entre deux composantes de la matrice par `\Cdots`, `\Vdots`, etc. (voir p. 32).

  L'extension nicematrix prédéfinit en fait les commandes `\hdottedline` et `\cdottedline` et la lettre « : » pour ces filets en pointillés.[26]

```
% déjà présent dans nicematrix.sty
\NiceMatrixOptions
  {
    custom-line =
      {
        letter = : ,
        command = hdottedline ,
        ccommand = cdottedline ,
        dotted
      }
  }
```

  Il est donc possible d'utiliser les commandes `\hdottedline` et `\cdottedline` pour tracer des filets horizontaux en pointillés.

[26] Néanmoins, l'utilisateur peut écraser ces définitions de `\hdottedline`, `\cdottedline` et de « : » avec `custom-line` s'il le souhaite (par exemple pour les remplacer par des lignes en tiretés).

```
\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 \\
\hdottedline
6 & 7 & 8 & 9 & 10 \\
\cdottedline{1,4-5}
11 & 12 & 13 & 14 & 15
\end{pNiceMatrix}
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{pmatrix}$$

Dans les environnements avec un préambule explicite (comme `{NiceTabular}`, `{NiceArray}`, etc.), il est possible de dessiner un trait vertical en pointillés avec le spécificateur « : ».

```
\begin{pNiceArray}{cccc:c}
1 & 2 & 3 & 4 & 5 \\
6 & 7 & 8 & 9 & 10 \\
11 & 12 & 13 & 14 & 15
\end{pNiceArray}
```

$$\left(\begin{array}{cccc:c} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{array}\right)$$

Comme on l'a dit, les clés précédentes peuvent être utilisés dans l'argument optionnel d'une commande `\Hline` individuelle ou dans l'argument optionnel d'un spécificateur « | » dans un préambule d'environnement (par exemple pour `{NiceTabular}`).
Mais dans ces cas-là, il est aussi possible d'utiliser deux clés supplémentaires, `start` et `end`, qui indiquent les numéros de rangées ou de colonnes des extrémités du filet.

```
\begin{NiceTabular}{cc|[color=blue,start=2]ccc}
un & deux & trois & quatre
\Hline[start=2,end=3]
cinq & six & sept & huit \\
neuf & dix & onze & douze
\end{NiceTabular}
```

un	deux	trois	quatre
cinq	six	sept	huit
neuf	dix	onze	douze

6 Les couleurs de fond des rangées et des colonnes

6.1 Utilisation de colortbl

Rappelons que l'extension colortbl peut être chargée directement par `\usepackage{colortbl}` ou en chargeant l'extension xcolor avec l'option `table` : `\usepackage[table]{xcolor}`.

Il y a néanmoins deux inconvénients :

- L'extension colortbl patche array, ce qui entraîne des incompatibilités (par exemple avec la commande `\hdotsfor`).
- L'extension colortbl construit le tableau rangée par rangée, en alternant rectangles colorés, filets et contenu des cases. Le PDF résultant déroute certains lecteurs de PDF et on a parfois des artefacts d'affichage.
  - Certains filets semblent disparaître. Ce phénomène est dû au fait que les lecteurs de PDF donnent souvent la priorité aux éléments graphiques qui ont été tracés postérieurement (conformément à l'esprit du « modèle du peintre » de PostScript et PDF). De ce point de vue, MuPDF (qui est utilisé par exemple par SumatraPDF) donne de meilleurs résultats que Adobe Reader.
  - Une fine ligne blanche semble apparaître entre deux cases de même couleur. Ce phénomène se produit quand chaque case est coloriée avec sa propre instruction `fill` (opérateur `fill` de PostScript noté `f` en PDF). C'est le cas avec colortbl avec lequel chaque case est coloriée individuellement, même si on utilise `\columncolor` ou `\rowcolor`.
    Concernant ce phénomène, Adobe Reader donne de meilleurs résultats que MuPDF (les versions récentes de MuPDF semblent avoir résolu ce problème).

  L'extension nicematrix propose des outils qui permettent d'éviter ces inconvénients.

6.2 Les outils de nicematrix dans le \CodeBefore

L'extension nicematrix propose des outils pour tracer d'abord les rectangles colorés, puis le contenu des cases et les filets. Cette manière de faire est plus dans l'esprit du « modèle du peintre » des formats PostScript et PDF et convient donc mieux aux lecteurs de PDF. L'inconvénient est qu'elle nécessite plusieurs compilations successives.[27]

L'extension nicematrix fournit une clé code-before pour du code qui sera exécuté avant le tracé du tableau. Une syntaxe alternative est proposée : on peut placer le contenu de ce code-before entre les mots-clés \CodeBefore et \Body juste au début de l'environnement.

```
\begin{pNiceArray}{preamble}
\CodeBefore [options]
  instructions du code-before
\Body
  contenu de l'environnement
\end{pNiceArray}
```

L'argument optionnel entre crochets est une liste de couples *clé=valeur* q
fur et à mesure (les clés disponibles sont create-cell-nodes, sub-matr
delimiters/color).

De nouvelles commandes sont disponibles dans ce \CodeBefore : \cellco
\rowcolor, \columncolor, \rowcolors, \rowlistcolors, \chessboardcol
Les noms de certaines de ces commandes sont inspirés des noms des comma

Ces commandes ne colorient pas les cases qui se trouvent dans les « coins »
utilisée. La description de cette clé a été faite p. 15.

Ces commandes respectent les coins arrondis si la clé rounded-corners (
p. 50) a été utilisée.

Toutes ces commandes acceptent un argument optionnel, entre crochets et e
argument optionel peut contenir deux éléments (séparés par une virgule) :

- le modèle colorimétrique (RGB, rgb, HTML, etc.) comme spécifié par l'ex
- une spécification d'opacité selon la forme opacity = *valeur*.[29]

On détaille maintenant ces différentes commandes.

- La commande \cellcolor tient son nom de la commande \cellcolo

  Elle prend en arguments obligatoires une couleur et une liste de cases
  est le numéro de rangée et j le numéro de colonne. Malgré son nom, ell
  rangée avec la syntaxe i- ou bien une colonne avec la syntaxe -j.

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \cellcolor[HTML]{FFFF88}{3-1,2-2,-3}
\Body
a & b & c \\
e & f & g \\
h & i & j \\
\end{NiceTabular}
```

a	b	c
e	f	g
h	i	j

[27] Si vous utilisez Overleaf, Overleaf effectue automatiquement un nombre de compilations suffisant (en utilisant latexmk).

[28] On pourra remarquer que, dans le \CodeBefore, des nœuds PGF-TikZ de la forme (i-|j) correspondant à la position des filets éventuels sont également accessibles : cf. p. 58.

[29] Attention : cette fonctionnalité génère des instructions de transparence dans le PDF résultant et certains lecteurs de PDF n'acceptent pas la transparence. L'application de l'opacité est faite par \pgfsetfillopacity.

- La commande `\rectanglecolor` prend t
les deux suivants fournissent la case en h

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \rectanglecolor{blue!15}{2-2}{3-3
\Body
a & b & c \\
e & f & g \\
h & i & j \\
\end{NiceTabular}
```

- La commande `\arraycolor` prend en argu
(sauf les éventuelles rangées et colonnes ex
cas particulier de la commande `\rectang`

- La commande `\chessboardcolors` pren
les cases en quinconces avec les deux coul

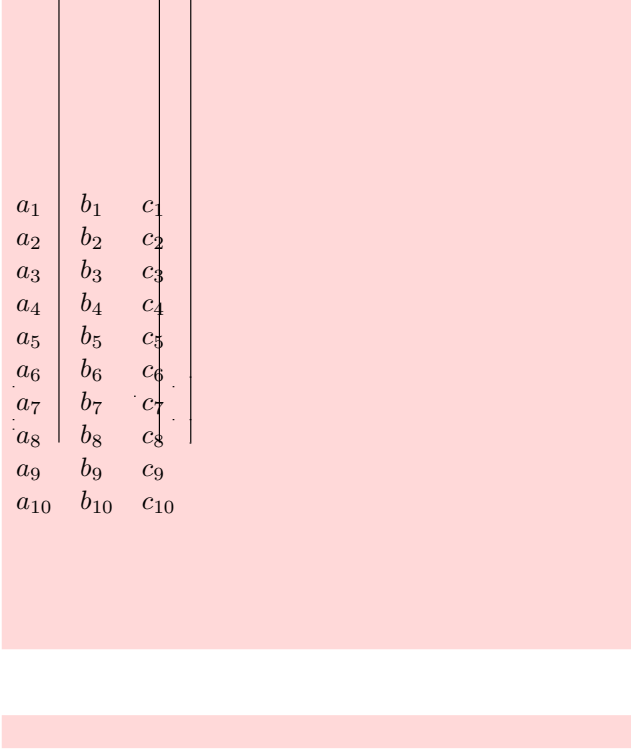
```
$\begin{pNiceMatrix}[r,margin]
\CodeBefore
  \chessboardcolors{red!15}{blue!15
\Body
1 & -1 & 1 \\
-1 & 1 & -1 \\
1 & -1 & 1
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix}$$

On a utilisé la clé `r` qui impose que toutes les colonnes soient alignées à droite (cf. p. 51).

- La commande `\rowcolor` doit son nom à la commande `\rowcolor` de colortbl. Son premier argument obligatoire est la couleur et le deuxième est une liste de numéros de rangées ou bien d'intervalles de rangées sous la forme *a*-*b* (un intervalle de la forme *a*- représente toutes les rangées à partir de la rangée *a*).

```
$\begin{NiceArray}{lll}[hvlines]
\CodeBefore
  \rowcolor{red!15}{1,3-5,8-}
\Body
a_1 & b_1 & c_1 \\
a_2 & b_2 & c_2 \\
a_3 & b_3 & c_3 \\
a_4 & b_4 & c_4 \\
a_5 & b_5 & c_5 \\
a_6 & b_6 & c_6 \\
a_7 & b_7 & c_7 \\
a_8 & b_8 & c_8 \\
a_9 & b_9 & c_9 \\
a_{10} & b_{10} & c_{10} \\
\end{NiceArray}$
```



- La commande `\columncolor` doit son nom à la commande `\columncolor` de colortbl. Sa syntaxe est similaire à celle de `\rowcolor`.

- La commande `\rowcolors` (avec un *s*) doit son nom à la commande `\rowcolors` de colortbl. Le *s* rappelle qu'il y a deux couleurs. Elle colorie alternativement les rangées avec les deux couleurs à partir de la rangée dont le numéro est donné en premier argument (obligatoire), comme le fait la commande `\rowcolors` de xcolor. L'un des deux arguments de couleur peut être vide (et alors aucune couleur n'est appliquée dans les rangées correspondantes).

  En fait, le premier argument (obligatoire) peut, plus généralement, contenir une liste d'intervalles correspondant à l'ensemble des rangées sur lesquelles portera l'effet de `\rowcolors` (un intervalle de la forme i désigne en fait l'intervalle constitué de toutes les rangées du tableau à partir de la rangée i).

  La commande `\rowcolors` accepte une liste de couples *clé=valeur* comme argument optionnel en dernière position (l'argument optionnel en première position correspond à l'espace colorimétrique). Les clés disponibles sont `cols`, `restart` et `respect-blocks`.

  - La clé `cols` décrit un ensemble de colonnes sur lesquelles portera l'effet de `\rowcolors`. Cet ensemble de colonnes est une liste d'intervalles de la forme i-j (où i et j peuvent être remplacés par `*`).
  - Avec la clé `restart`, chacun des intervalles de rangées spécifié par le premier argument de `\rowcolors` recommence avec la même couleur.[30]
  - Avec la clé `respect-blocks`, qui est de type booléen, les « rangées » colorées alternativement peuvent s'étendre sur plusieurs rangées réelles du tableau pour englober les blocs (créés par la commande `\Block` : cf. p. 5).

[30] Autrement, la couleur d'une rangée ne dépend que de la parité de son numéro absolu.

```
\begin{NiceTabular}{clr}[hvlines]
\CodeBefore
  \rowcolors[gray]{2}{0.8}{}[cols=2-3,restart]
\Body
\Block{1-*}{Résultats} \\
\Block{2-1}{A}& Pierre & 12 \\
              & Jacques & 8 \\
\Block{4-1}{B}& Stéphanie & 18 \\
              & Amélie & 20 \\
              & Henri & 14 \\
              & Estelle & 15
\end{NiceTabular}
```

```
\begin{NiceTabular}{lr}[hvlines]
\CodeBefore
  \rowcolors{1}{blue!10}{}[respect-blocks]
\Body
\Block{2-1}{Pierre}    & 12 \\
                       & 13 \\
Jacques                &  8 \\
\Block{3-1}{Stéphanie} & 18 \\
                       & 17 \\
                       & 15 \\
Amélie                 & 20 \\
Henri                  & 14 \\
\Block{2-1}{Estelle}   & 15 \\
                       & 19
\end{NiceTabular}
```

- L'extension nicematrix propose aussi une commande `\rowlistcolors`. Cette commande généralise la commande `\rowcolors` : au lieu de prendre deux arguments successifs pour les couleurs elle prend un seul argument qui est une *liste* de couleurs séparées par des virgules. Dans cette liste, le symbole `=` représente une couleur identique à la précédente.

```
\begin{NiceTabular}{c}
\CodeBefore
  \rowlistcolors{1}{red!15,blue!15,green!15}
\Body
Mathilde \\
Pierre \\
Paul \\
Amélie \\
Jacques \\
Antoine \\
Stéphanie \\
\end{NiceTabular}
```

On peut aussi utiliser dans la commande `\rowlistcolors` une série de couleurs définie par la commande `\definecolorseries` de xcolor (et initialisée avec `\resetcolorseries`[31]).

[31] Pour l'initialisation, on a utilisé dans l'exemple qui suit le compteur LaTeX `iRow` (qui correspond en interne au compteur TeX `\c@iRow`) qui, quand il est utilisé dans le `\CodeBefore` (ou le `\CodeAfter`) désigne le nombre de rangées du tableau : cf p. 53. Cela permet un ajustement de la gradation des couleurs à la taille du tableau.

```
\begin{NiceTabular}{c}
\CodeBefore
   \definecolorseries{Bl
   \resetcolorseries{\va
   \rowlistcolors{1}{Blu
\Body
Mathilde \\
Pierre \\
Paul \\
Amélie \\
Jacques \\
Antoine \\
Stéphanie \\
\end{NiceTabular}
```

On rappelle que toutes les comm
cases qui sont dans les « coins ». D
considérer le coin *north east* (NE)

```
\begin{NiceTabular}{cccccc}
 [corners=NE,margin,hvlines,
\CodeBefore
  \rowlistcolors{1}{blue!15,
\Body
  & 0 & 1 & 2 & 3 & 4 & 5 &
0 & 1 \\
1 & 1 & 1 \\
2 & 1 & 2 & 1 \\
3 & 1 & 3 & 3 & 1 \\
4 & 1 & 4 & 6 & 4 & 1 \\
5 & 1 & 5 & 10 & 10 & 5 & 1
6 & 1 & 6 & 15 & 20 & 15 & 6
\end{NiceTabular}
```

L'exemple précédent utilise les clé
rangées et colonnes « extérieures »
Comme on le voit, *par défaut*, les c
dans ces rangées et colonnes « ext
Mais on peut *quand même* colorie
dentes les numéros explicites de c
Dans l'exemple suivant, on deman
colonne » et qui existe du fait de l

```
\begin{NiceTabular}{cccccc}[
\CodeBefore
  \rowlistcolors{1}{blue!15,
  \columncolor{red!15}{0}
\Body
  & 0 & 1 & 2 & 3 & 4 & 5 &
0 & 1 \\
1 & 1 & 1 \\
2 & 1 & 2 & 1 \\
3 & 1 & 3 & 3 & 1 \\
4 & 1 & 4 & 6 & 4 & 1 \\
5 & 1 & 5 & 10 & 10 & 5 & 1
6 & 1 & 6 & 15 & 20 & 15 & 6 & 1 \\
\end{NiceTabular}
```

	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

On remarquera que ces commandes sont compatibles avec les commandes de bookt
\midrule, \bottomrule, etc). Néanmoins, l'extension booktabs n'est *pas* chargée pa

```
\begin{NiceTabular}{lSSSS}
\CodeBefore
  \rowcolor{red!15}{1-2}
  \rowcolors{3}{blue!15}{}
\Body
\toprule
\Block[C]{2-1}{Produit} &
\Block{1-3}{dimensions (cm)} & & &
\Block{2-1}{\rotate Prix} \\
\cmidrule(rl){2-4}
 & L & l & h \\
\midrule
petit    & 3   & 5.5  & 1   & 30    \\
moyen    & 5.5 & 8    & 1.5 & 50.5  \\
premium  & 8.5 & 10.5 & 2   & 80    \\
extra    & 8.5 & 10   & 1.5 & 85.5  \\
spécial  & 12  & 12   & 0.5 & 70    \\
\bottomrule
\end{NiceTabular}
```

Produit	dimensions (cm)			Prix
	L	l	h	
petit	3	5.5	1	30
moyen	5.5	8	1.5	50.5
premium	8.5	10.5	2	80
extra	8.5	10	1.5	85.5
spécial	12	12	0.5	70

On a utilisé le type de colonne S de siunitx (qu'il faut avoir charg

On peut aussi, dans le \CodeBefore, utiliser les commandes \Em nn et \EmptyRow. La commande \EmptyColumn prend en argument une liste de numéro onnes et impose qu'aucun coloriage ni tracé de filets n'aura lieu dans les colonnes correspo La commande \EmptyRow est similaire.

```
\begin{NiceTabular}{ccccc}[hvlines,no-cell-nodes]
\CodeBefore
  \rowcolor{blue!15}{1}
  \EmptyColumn{3}
\Body
  un & deux && trois & quatre \\
  un & \Block{}{deux\\ lignes} && trois & quatre \\
\end{NiceTabular}
```

un	deux		trois	quatre
un			trois	quatre

6.3 Outils de coloriage en tableau

L'extension nicematrix propose aussi des commandes de coloriage à utiliser directement dans le tableau (comme celles de l'extension colortbl).
Les commandes sont les suivantes (les trois premières sont inspirées par les commandes similaires de colortbl).

- \cellcolor qui colorie la case courante[32] ;
- \rowcolor à utiliser dans une case et qui colorie le reste de la rangée[33] ;
- \columncolor à utiliser dans le préambule du tableau de la même manière que la commande homonyme de colortbl (néanmoins, contrairement à la commande \columncolor de colortbl, celle de nicematrix peut apparaître à l'intérieur d'une autre commande, elle-même utilisée dans le préambule ; en revanche, elle ne prend pas les deux arguments optionnels à la fin entre crochets pour du débord comme la commande de colortbl) ;

[32] Cette commande \cellcolor supprimera les espaces qui la suivent (ce que ne fait pas la commande \cellcolor de colortbl). De plus, si on définit une fonction au-dessus de \cellcolor, il faudra une fonction protégée au sens de TeX (alors que si c'était la commande \cellcolor de colortbl, il faudrait au contaire une fonction *fully expandable*).
[33] Si vous souhaitez une commande pour colorier les n rangées suivantes, considérez la commande \RowStyle et sa clé fill, p. 27.

- `\rowcolors` qui prend pour arguments deux couleurs et colorie la suite du tableau avec ces deux couleurs (à utiliser après un éventuel `\hline`, `\Hline` ou `\toprule`!) ;
- `\rowlistcolors` qui prend pour argument une liste de couleurs et colorie la suite du tableau avec ces couleurs.[34]

Ces commandes sont compatibles avec les commandes pour les *overlays* de Beamer (comme `\only`, etc.)

```
\NewDocumentCommand { \Blue } { } {\columncolor{blue!15}}
\begin{NiceTabular}{>{\Blue}c>{\Blue}cc}
\toprule
\rowcolor{red!15}
Nom & Prénom & Année de naissance \\
\midrule
Achard & Jacques & 5 juin 1962 \\
Lefebvre & Mathilde & 23 mai 1988 \\
Vanesse & Stéphanie & 30 octobre 1994 \\
Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}
```

Nom	Prénom	Année de naissance
Achard	Jacques	5 juin 1962
Lefebvre	Mathilde	23 mai 1988
Vanesse	Stéphanie	30 octobre 1994
Dupont	Chantal	15 janvier 1998

Chaque utilisation de `\rowlistcolors` (et de `\rowcolors` qui en est un cas particulier) met un terme aux éventuels schémas[35] de coloriage en cours qui auraient été spécifiés par une commande `\rowlistcolors` précédente.
En particulier, on peut engager un coloriage des rangées avec `\rowlistcolors{...}` et l'arrêter par un `\rowlistcolors{}` avec argument vide.

```
\begin{NiceTabular}{c}[hvlines]
un \\
deux \\
\rowlistcolors{red!15}
trois \\
quatre \\
cinq \\
\rowlistcolors{}
six \\
sept \\
\end{NiceTabular}
```

un
deux
trois
quatre
cinq
six
sept

6.4 La couleur spécial « nocolor »

L'extension nicematrix propose la couleur spéciale `nocolor` utilisable dans toutes les commandes de coloriage fournies par nicematrix (dans le `\CodeBefore` ou bien dans le tableau proprement dit).
Les cases marquées par cette couleur ne seront pas coloriées, quelles que soient les autres commandes de coloriage qui auraient pu s'appliquer à ces cases.
La couleur `nocolor` fournit donc un moyen commode de faire des exceptions à l'action d'une commande de coloriage générale.

[34] Quand la commande `\rowlistcolors` (ou la commande `\rowcolors`) est utilisée dans une case de la colonne j, le coloriage ne s'applique que sur les colonnes au-delà de j (à dessein).
[35] On a écrit *schémas* au pluriel car on peut avoir plusieurs schémas en cours s'ils portent sur des colonnes différentes.

7 La commande \RowStyle

La commande `\RowStyle` prend en argument des instructions de mise en forme qui seront appliquées à chacune des cases restantes sur la rangée en cours.

Elle prend aussi en premier argument optionnel, entre crochets, une liste de couples *clé=valeur*.

- La clé `nb-rows` indique le nombre de rangées consécutives concernées par les spécifications de cette commande (une valeur `*` signifie que toutes les rangées restantes seront concernées).
- Les clés `cell-space-top-limit`, `cell-space-bottom-limit` et `cell-space-limits` sont disponibles avec le même effet que les clés globales de même nom (cf. p. 3).
- La clé `fill` (alias : `rowcolor`) fixe la couleur de fond et la clé `opacity`[36] l'opacité de cette couleur de fond. Si la clé `rounded-corners` est utilisée, ce fond aura des coins arrondis.
- La clé `color` fixe la couleur du texte.[37]
- La clé `bold` impose des caractères gras aux éléments de la rangée, qu'ils soient en mode texte ou bien en mode mathématique.

```
\begin{NiceTabular}{cccc}
\hline
\RowStyle[cell-space-limits=3pt]{\rotate}
premier & deuxième & troisième & quatrième \\
\RowStyle[nb-rows=2,color=white,fill=blue!50]{\sffamily}
1 & 2 & 3 & 4 \\
I & II & III & IV
\end{NiceTabular}
```

premier	deuxième	troisième	quatrième
1	2	3	4
I	II	III	IV

La commande `\rotate` est présentée p. 52.

8 La largeur des colonnes

8.1 Techniques de base

Dans les environnements avec un préambule explicite (comme `{NiceTabular}`, `{NiceArray}`, etc.), il est possible de fixer la largeur d'une colonne avec les lettres classiques `w`, `W`, `p`, `b` et `m` de l'extension array (qui est chargée par nicematrix).

```
\begin{NiceTabular}{W{c}{2cm}cc}[hvlines]
Paris  & New York & Madrid \\
Berlin & London   & Roma   \\
Rio    & Tokyo    & Oslo
\end{NiceTabular}
```

Paris	New York	Madrid
Berlin	London	Roma
Rio	Tokyo	Oslo

Dans les environnements de nicematrix, il est aussi possible de fixer la largeur *minimale* de toutes les colonnes (à l'exception des éventuelles colonnes extérieures : cf. p. 30) directement avec l'option `columns-width`.

[36] Attention : cette clé génère des instructions de transparence dans le PDF résultant et certains lecteurs de PDF n'acceptent pas la transparence.

[37] La clé `color` utilise la commande `\color` mais insère aussi une instruction `\leavevmode` devant. Cela évite un espace vertical parasite dans les cases qui correspondent à des colonnes de type `p`, `b`, `m`, et `X` (qui débutent en mode vertical de LaTeX). Pour les colonnes de type `V` (de varwidth), cela ne suffit malheureusement pas sauf si on utilise LuaLaTeX avec luacolor (cf. question 460489 sur TeX StackExchange).

```
$\begin{pNiceMatrix}[columns-width = 1cm]
1  & 12 & -123 \\
12 & 0  & 0    \\
4  & 1  & 2
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & 12 & -123 \\ 12 & 0 & 0 \\ 4 & 1 & 2 \end{pmatrix}$$

Notez que l'espace inséré entre deux colonnes (égal à 2 `\tabcolsep` dans `{NiceTabular}` et à 2 `\arraycolsep` dans les autres environnements) n'est pas supprimé (il est évidemment possible de le supprimer en mettant `\tabcolsep` ou `\arraycolsep` à 0 avant).

Il est possible de donner la valeur spéciale `auto` à l'option `columns-width` : toutes les colonnes du tableau auront alors une largeur égale à la largeur de la case la plus large du tableau.[38]

```
$\begin{pNiceMatrix}[columns-width = auto]
1  & 12 & -123 \\
12 & 0  & 0    \\
4  & 1  & 2
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & 12 & -123 \\ 12 & 0 & 0 \\ 4 & 1 & 2 \end{pmatrix}$$

Sans surprise, il est possible de fixer la largeur minimale de toutes les colonnes de tous les tableaux dans une certaine portion de document avec la commande `\NiceMatrixOptions`.

```
\NiceMatrixOptions{columns-width=10mm}
$\begin{pNiceMatrix}
a & b \\ c & d
\end{pNiceMatrix}
=
\begin{pNiceMatrix}
1   & 1245 \\ 345 & 2
\end{pNiceMatrix}$
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 1245 \\ 345 & 2 \end{pmatrix}$$

Mais il est aussi possible de fixer une zone dans laquelle toutes les matrices auront leurs colonnes de la même largeur, égale à la largeur de la case la plus large de toutes les matrices de la zone. Cette construction utilise l'environnement `{NiceMatrixBlock}` avec l'option `auto-columns-width`[39]. L'environnement `{NiceMatrixBlock}` n'a pas de rapport direct avec la commande `\Block` présentée précédemment dans ce document (cf. p. 5).

```
\begin{NiceMatrixBlock}[auto-columns-width]
$\begin{array}{c}
\begin{bNiceMatrix}
 9 & 17 \\ -2 & 5
 \end{bNiceMatrix} \\ \\
\begin{bNiceMatrix}
 1   & 1245345 \\  345 & 2
\end{bNiceMatrix} \\
\end{array}$
\end{NiceMatrixBlock}
```

$$\begin{bmatrix} 9 & 17 \\ -2 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1245345 \\ 345 & 2 \end{bmatrix}$$

8.2 Les colonnes V de varwidth

Rappelons d'abord le fonctionnement d'un environnement `{varwidth}` de l'extension éponyme varwidth. Un tel environnement est similaire à l'environnement classique `{minipage}` mais la largeur indiquée (en argument) n'est que la largeur *maximale* de la boîte créée. Dans le cas général, la largeur d'une boîte `{varwidth}` est la largeur naturelle de son contenu.

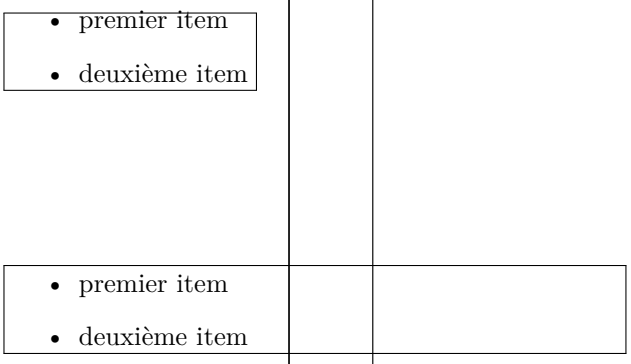
Cela est illustré avec les exemples suivants :

[38] Le résultat est atteint dès la première compilation (mais PGF-TikZ écrivant des informations dans le fichier `aux`, un message demandant une deuxième compilation apparaîtra).

[39] Pour le moment, c'est le seul usage de l'environnement `{NiceMatrixBlock}` mais il pourrait y en avoir davantage dans le futur.

```
\fbox{%
\begin{varwidth}{8cm}
\begin{itemize}
\item premier item
\item deuxième item
\end{itemize}
\end{varwidth}}

\fbox{%
\begin{minipage}{8cm}
\begin{itemize}
\item premier item
\item deuxième item
\end{itemize}
\end{minipage}}
```



L'extension varwidth définit également le type de colonne `V`. Une colonne `V{`⟨*dim*⟩`}` encapsule toutes ses cases dans une `{varwidth}` d'argument ⟨*dim*⟩ (et effectue quelques réglages supplémentaires).

Lorsque l'extension varwidth est chargée, ces colonnes `V` de varwidth sont prises en charge par nicematrix.

```
\begin{NiceTabular}[corners=NW,hvlines]{V{3cm}V{3cm}V{3cm}}
& un texte & un très très très très long texte \\
un très très très très long texte \\
un très très très très long texte
\end{NiceTabular}
```

	un texte	un très très très très long texte
un très très très très long texte		
un très très très très long texte		

Dans le cadre de nicematrix, l'un des intérêts des colonnes de type `V` par rapport aux colonnes de type `p`, `m` ou `b` est que, pour les cases d'une telle colonne, le nœud PGF-TikZ créé pour le contenu d'une telle case a une largeur ajustée au contenu de la case en question : cf. p. 56.

Les colonnes `V` de nicematrix acceptent les clés `t`, `p`, `m`, `b`, `l`, `c` et `r` proposées par les colonnes `X` : voir leur description à la section 8.3, p. 30.

Remarquons que l'extension varwidth a quelques problèmes (au moins dans sa version 0.92). Par exemple, avec LuaLaTeX, elle ne fonctionne pas si le contenu commence par une instruction `\color`. De plus, varwidth doit être chargée après l'extension array (elle-même chargée par nicematrix).

8.3 Les colonnes X

L'environnement `{NiceTabular}` propose aussi des colonnes `X` similaires à celles proposées par l'environnement `{tabularx}` de l'extension éponyme.
La valeur requise par la largeur du tableau peut être passée en argument de la clé `width` (dans `{NiceTabular}` ou dans `\NiceMatrixOptions`). La valeur initiale de ce paramètre est `\linewidth` (et non `\textwidth`).
Pour se rapprocher davantage de l'environnement `{tabularx}`, nicematrix propose aussi un environnement `{NiceTabularX}` avec une syntaxe similaire à celle de `{tabularx}`, c'est-à-dire que la largeur voulue pour le tableau est spécifiée en premier argument (obligatoire).

Comme avec les extensions tabu[40] et tabularray, le spécificateur `X` accepte entre crochets un argument optionnel qui est une liste de clés.

- On peut spécifier un poids pour la colonne en mettant directement un nombre positif comme argument du spécificateur `X`. Par exemple, une colonne `X[2]` aura une largeur double de celle d'une colonne `X` (qui a un poids de 1).[41]
- On peut spécifier l'alignement horizontal avec une des lettres `l`, `c` et `r` (qui insèrent respectivement `\raggedright`, `\centering` et `\raggedleft` suivi de `\arraybackslash`).[42]
- On peut spécifier l'alignement vertical avec l'une des lettres `t` (alias `p`), `m` et `b` (qui construisent respectivement des colonnes de types `p`, `m` et `b`). La valeur initiale est `t`.
- Il est possible d'utiliser la clé `V` dans une colonne de type `X`. Quand cette clé est utilisée, la colonne `X` se comporte en fait comme une colonne `V` de l'extension varwidth (que l'utilisateur doit avoir chargée), ce qui fait que la largeur de la colonne calculée par le processus `X` devient la largeur *maximale* de la colonne.

```
\begin{NiceTabular}[width=9cm]{X[c,m]X[0.5,c,m]}[hvlines]
Un texte relativement long qui tient sur plusieurs lignes. &
Un texte relativement long qui tient sur plusieurs lignes. \\
Un texte plus court. & Un texte plus court.
\end{NiceTabular}
```

Un texte relativement long qui tient sur plusieurs lignes.	Un texte relativement long qui tient sur plusieurs lignes.
Un texte plus court.	Un texte plus court.

9 Les rangées et colonnes extérieures

Les environnements de nicematrix permettent de composer des rangées et des colonnes « extérieures » grâce aux options `first-row`, `last-row`, `first-col` et `last-col`. C'est particulièrement intéressant pour les matrices (mathématiques).
Si elle est présente, la « première rangée » (extérieure) est numérotée par 0 (et non 1). Il en est de même pour la « première colonne ».

```
$\begin{pNiceMatrix}[first-row,last-row,first-col,last-col,nullify-dots]
       & C_1    & \Cdots &        & C_4    &        \\
L_1    & a_{11} & a_{12} & a_{13} & a_{14} & L_1    \\
\Vdots & a_{21} & a_{22} & a_{23} & a_{24} & \Vdots \\
       & a_{31} & a_{32} & a_{33} & a_{34} &        \\
L_4    & a_{41} & a_{42} & a_{43} & a_{44} & L_4    \\
       & C_1    & \Cdots &        & C_4    &
\end{pNiceMatrix}$
```

$$\begin{array}{c} \\ L_1 \\ \\ \\ L_4 \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & & & C_4 \end{array} \\ \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \\ \begin{array}{cccc} C_1 & & & C_4 \end{array} \end{array} \begin{array}{c} \\ L_1 \\ \\ \\ L_4 \\ \\ \end{array}$$

[40] L'extension tabu est maintenant considérée comme obsolète.
[41] Les valeurs négatives pour les poids, comme proposées par tabu (maintenant obsolète), ne sont *pas* prises en charge par nicematrix. Si une telle valeur est utilisée, une erreur sera levée.
[42] En fait, quand ragged2e est chargée, ce sont les commandes `\RaggedRight`, `\Centering` et `\RaggedLeft` de ragged2e qui sont utilisées, pour un meilleur résultat.

Les lignes pointillées ont été tracées avec les outils qui seront présentés p. 32.

Il y a plusieurs remarques à formuler.

- Si on utilise un environnement avec préambule explicite (`{NiceTabular}`, `{NiceArray}` ou l'une de ses variantes), on ne doit pas mettre dans ce préambule de spécification de colonne pour les éventuelles première et dernière colonne : ce sera automatiquement (et nécessairement) une colonne `r` pour la première colonne et une colonne `l` pour la dernière.[43]
- On peut se demander comment nicematrix détermine le nombre de rangées et de colonnes nécessaires à la composition de la « dernière rangée » et de la « dernière colonne ».
  - Dans le cas d'un environnement avec préambule, comme `{NiceTabular}` ou `{pNiceArray}`, le nombre de colonnes se déduit évidemment du préambule.
  - Dans le cas où l'option `light-syntax` (cf. p. 54) est utilisée, nicematrix profite du fait que cette option nécessite de toutes manières le chargement complet du contenu de l'environnement (d'où l'impossibilité de mettre du verbatim dans ce cas-là) avant composition du tableau. L'analyse du contenu de l'environnement donne le nombre de rangées et de colonnes.
  - Dans les autres cas, nicematrix détermine le nombre de rangées et de colonnes à la première compilation et l'écrit dans le fichier `aux` pour pouvoir l'utiliser à la compilation suivante.

    Néanmoins, il est possible de donner le numéro de la dernière rangée et le numéro de la dernière colonne en arguments des options `last-row` *et* `last-col`*, ce qui permettra d'accélérer le processus complet de compilation.* C'est ce que nous ferons dans la suite.

On peut contrôler l'apparence de ces rangées et colonnes avec les options `code-for-first-row`, `code-for-last-row`, `code-for-first-col` et `code-for-last-col`. Ces options sont des listes de tokens qui seront insérées au début de chaque case de la rangée ou de la colonne considérée.

```
\NiceMatrixOptions{code-for-first-row = \color{red},
                   code-for-first-col = \color{blue},
                   code-for-last-row = \color{green},
                   code-for-last-col = \color{magenta}}
$\begin{pNiceArray}{cc|cc}[first-row,last-row=5,first-col,last-col,nullify-dots]
       & C_1    & \Cdots &        & C_4    &        \\
L_1    & a_{11} & a_{12} & a_{13} & a_{14} & L_1    \\
\Vdots & a_{21} & a_{22} & a_{23} & a_{24} & \Vdots \\
\hline
       & a_{31} & a_{32} & a_{33} & a_{34} &        \\
L_4    & a_{41} & a_{42} & a_{43} & a_{44} & L_4    \\
       & C_1    & \Cdots &        & C_4    &
\end{pNiceArray}$
```

$$\begin{array}{c} \\ L_1 \\ \\ \\ L_4 \\ \\ \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & & & C_4 \end{array} \\ \left(\begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & a_{14} & L_1 \\ a_{21} & a_{22} & a_{23} & a_{24} & \\ \hline a_{31} & a_{32} & a_{33} & a_{34} & \\ a_{41} & a_{42} & a_{43} & a_{44} & L_4 \end{array}\right) \\ \begin{array}{cccc} C_1 & & & C_4 \end{array} \end{array}$$

Remarques

- Comme on peut le voir dans l'exemple précédent, les filets horizontaux et verticaux ne s'étendent pas dans les rangées et colonnes extérieures. Cette remarque s'applique aussi aux filets définis par les outils de personnalisation de nicematrix (cf. la clé `custom-line` p. 17).

[43] Si on souhaite une colonne extérieure avec un autre type d'alignement, on aura intérêt à considérer la commande `\SubMatrix` disponible dans le `\CodeAfter` et le `\CodeBefore` (cf. p. 41) ou bien voir à insérer des délimiteurs directement dans le préambule du tableau (cf. p. 39).

- Une spécification de couleur présente dans `code-for-first-row` s'applique à une ligne pointillée tracée dans cette « première rangée » (sauf si une valeur a été donnée à `xdots/color`). Idem pour les autres.
- Sans surprise, une éventuelle option `columns-width` (décrite p. 28) ne s'applique pas à la « première colonne » ni à la « dernière colonne ».
- Pour des raisons techniques, il n'est pas possible d'utiliser l'option de la commande `\\` après la « première rangée » ou avant la « dernière rangée ». Le placement des délimiteurs serait erroné. Pour contourner cette restriction, on pourra envisager d'utiliser la commande `\SubMatrix` dans le `\CodeAfter` (cf. p. 41).

10 Les lignes en pointillés continues

À l'intérieur des environnements de l'extension nicematrix, de nouvelles commandes sont définies : `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` et `\Iddots`. Ces commandes sont conçues pour être utilisées à la place de `\dots`, `\cdots`, `\vdots`, `\ddots` et `\iddots`.[44]

Chacune de ces commandes doit être utilisée seule dans la case du tableau et elle trace une ligne en pointillés entre les premières cases non vides[45] situées de part et d'autre de la case courante. Bien entendu, pour `\Ldots` et `\Cdots`, c'est une ligne horizontale ; pour `\Vdots`, c'est une ligne verticale et pour `\Ddots` et `\Iddots`, ce sont des lignes diagonales. On peut changer la couleur d'une ligne avec l'option `color`.[46]

```
\begin{bNiceMatrix}
a_1      & \Cdots &        & & a_1     \\
\Vdots   & a_2    & \Cdots & & a_2     \\
         & \Vdots & \Ddots[color=red] \\
\\
a_1      & a_2    &        & & a_n
\end{bNiceMatrix}
```

$$\begin{bmatrix} a_1 & & & a_1 \\ & a_2 & & a_2 \\ & & & \\ a_1 & a_2 & & a_n \end{bmatrix}$$

Pour représenter la matrice nulle, on peut choisir d'utiliser le codage suivant :

```
\begin{bNiceMatrix}
0      & \Cdots & 0      \\
\Vdots &        & \Vdots \\
0      & \Cdots & 0
\end{bNiceMatrix}
```

$$\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

On peut néanmoins souhaiter une matrice plus grande. Habituellement, dans un tel cas, les utilisateurs de LaTeX ajoutent une nouvelle ligne et une nouvelle colonne. Il est possible d'utiliser la même méthode avec nicematrix :

```
\begin{bNiceMatrix}
0      & \Cdots & \Cdots & 0      \\
\Vdots &        &        & \Vdots \\
\Vdots &        &        & \Vdots \\
0      & \Cdots & \Cdots & 0
\end{bNiceMatrix}
```

$$\begin{bmatrix} 0 & & & 0 \\ & & & \\ & & & \\ 0 & & & 0 \end{bmatrix}$$

Dans la première colonne de cet exemple, il y a deux instructions `\Vdots` mais, bien entendu, une seule ligne en pointillés sera tracée.

[44] La commande `\iddots`, définie par nicematrix, est une variante de `\ddots` avec les points allant vers le haut. Si mathdots est chargée, la version de mathdots est utilisée. Elle correspond à la commande `\adots` de unicode-math.

[45] La définition précise de ce qui est considéré comme une « case vide » est donnée plus loin (cf. p. 61).

[46] Il est aussi possible de changer la couleur de toutes ces lignes pointillées avec l'option `xdots/color` (*xdots* pour rappeler que cela s'applique à `\Cdots`, `\Ldots`, `\Vdots`, etc.) : cf. p. 36.

En fait, dans cet exemple, il aurait été possible de tracer la même matrice plus rapidement avec le codage suivant (parmi d'autres) :

```
\begin{bNiceMatrix}
0       & \Cdots &         & 0       \\
\Vdots  &        &         &         \\
        &        &         & \Vdots  \\
0       &        & \Cdots  & 0
\end{bNiceMatrix}
```

$$\begin{bmatrix} 0 & & 0 \\ & & \\ & & \\ 0 & & 0 \end{bmatrix}$$

Il y a aussi d'autres moyens de changer la taille d'une matrice. On pourrait vouloir utiliser l'argument optionnel de la commande `\\` pour l'espacement vertical et la commande `\hspace*` dans une case pour l'espacement horizontal.[47]
Toutefois, une commande `\hspace*` pourrait interférer dans la construction des lignes en pointillés. C'est pourquoi l'extension nicematrix fournit une commande `\Hspace` qui est une variante de `\hspace` transparente pour la construction des lignes en pointillés de nicematrix.

```
\begin{bNiceMatrix}
0       & \Cdots & \Hspace*{1cm} & 0       \\
\Vdots  &        &               & \Vdots  \\[1cm]
0       & \Cdots &               & 0
\end{bNiceMatrix}
```

$$\begin{bmatrix} 0 & & 0 \\ & & \\ & & \\ 0 & & 0 \end{bmatrix}$$

10.1 L'option nullify-dots

Considérons la matrice suivante qui a été composée classiquement avec l'environnement `{pmatrix}` de amsmath.

```
$A = \begin{pmatrix}
h & i & j & k & l & m \\
x &   &   &   &   & x
\end{pmatrix}$
```

$$A = \begin{pmatrix} h & i & j & k & l & m \\ x & & & & & x \end{pmatrix}$$

Si nous ajoutons des instructions `\ldots` dans la seconde rangée, la géométrie de la matrice est modifiée.

```
$B = \begin{pmatrix}
h & i & j & k & l & m \\
x & \ldots & \ldots & \ldots & \ldots & x
\end{pmatrix}$
```

$$B = \begin{pmatrix} h & i & j & k & l & m \\ x & \ldots & \ldots & \ldots & \ldots & x \end{pmatrix}$$

Par défaut, avec nicematrix, si nous remplaçons `{pmatrix}` par `{pNiceMatrix}` et `\ldots` par `\Ldots`, la géométrie de la matrice n'est pas changée.

```
$C = \begin{pNiceMatrix}
h & i & j & k & l & m \\
x & \Ldots & \Ldots & \Ldots & \Ldots & x
\end{pNiceMatrix}$
```

$$C = \begin{pmatrix} h & i & j & k & l & m \\ x & & & & & x \end{pmatrix}$$

On pourrait toutefois préférer la géométrie de la première matrice A et vouloir avoir la même géométrie avec une ligne en pointillés continue dans la seconde rangée. C'est possible en utilisant l'option `nullify-dots` (et une seule instruction `\Ldots` suffit).

```
$D = \begin{pNiceMatrix}[nullify-dots]
h & i & j & k & l & m \\
x & \Ldots & & & & x
\end{pNiceMatrix}$
```

$$D = \begin{pmatrix} h & i & j & k & l & m \\ x & & & & & x \end{pmatrix}$$

[47] Dans nicematrix, il faut utiliser `\hspace*` et non `\hspace` car nicematrix utilise array. Remarquons aussi que l'on peut également régler la largeur des colonnes en utilisant l'environnement `{NiceArray}` (ou une de ses variantes) avec une colonne de type `w` ou `W` : cf. p. 28