The package nicematrix*

F. Pantigny
fpantigny@wanadoo.fr

November 9, 2026

Abstract

The LaTeX package nicematrix provides new environments similar to the classical environments `{tabular}`, `{array}` and `{matrix}` of array and amsmath but with extended features.

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

Product	dimensions (cm)			Price
	L	l	h	
small	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
special	12	12	0.5	70

The package nicematrix is entirely contained in the file `nicematrix.sty`. This file may be put in the current directory or in a `texmf` tree. However, the best is to install nicematrix with a TeX distribution such as MiKTeX, TeX Live or MacTeX.

Remark
If you use LaTeX via Internet with, for example, Overleaf or TeXPage, you can upload the file `nicematrix.sty` in the repertory of your project in order to take full advantage of the latest version de nicematrix.[1]

This package can be used with `xelatex`, `lualatex`, `pdflatex` but also by the classical workflow `latex-dvips-ps2pdf` (or Adobe Distiller). *However, the file nicematrix.dtx of the present documentation should be compiled with LuaLaTeX.*

This package requires and **loads** the packages array, amsmath, pgfcore and the module shapes of PGF (tikz, which is a layer over PGF, is *not* loaded). The final user only has to load the package with `\usepackage{nicematrix}`.

The idea of nicematrix is to create PGF nodes under the cells and the positions of the rules of the tabular created by array and to use these nodes to develop new features. As usual with PGF, the coordinates of these nodes are written in the `aux` file to be used on the next compilation and that's why nicematrix may need **several compilations**[2]. One must not use the command `\nofiles` (which prevents the creation of the `aux` file).

Most features of nicematrix may be used without explicit use of PGF or TikZ (which, in fact, is not loaded by default).

A command `\NiceMatrixOptions` is provided to fix the options (the scope of the options fixed by this command is the current TeX group: they are semi-global).

*This document corresponds to the version 7.13 of nicematrix, at the date of 2026/11/09.

[1] The latest version of the file `nicematrix.sty` may be downloaded on the GitHub depot of nicematrix: `https://github.com/fpantigny/nicematrix/releases`

[2] If you use Overleaf, Overleaf will do automatically a sufficient number of compilations (by using `latexmk`).

Contents

1 The environments of this package **4**

2 The vertical space between the rows **4**

3 The vertical position of the arrays **5**

4 The blocks **6**
- 4.1 General case . . . 6
- 4.2 The mono-column blocks . . . 9
- 4.3 The mono-row blocks . . . 9
- 4.4 The mono-cell blocks . . . 9
- 4.5 Horizontal position of the content of the block . . . 10
- 4.6 Vertical position of the content of the block . . . 11
- 4.7 \\ and & in the blocks . . . 12
- 4.8 The key create-blocks-in-col . . . 13

5 The rules **14**
- 5.1 Some differences with the classical environments . . . 14
  - 5.1.1 The vertical rules . . . 14
  - 5.1.2 The command \cline . . . 15
- 5.2 The thickness and the color of the rules . . . 15
- 5.3 The tools of nicematrix for the rules . . . 15
  - 5.3.1 The keys hlines and vlines . . . 16
  - 5.3.2 The keys hvlines, hlines-except-borders and hvlines-except-borders . . . 17
  - 5.3.3 The (empty) corners . . . 17
  - 5.3.4 The command \diagbox . . . 18
  - 5.3.5 Commands for customized rules . . . 19
  - 5.3.6 The key fix-vertex . . . 23
  - 5.3.7 The key default-line . . . 24
  - 5.3.8 The command \FalseRow and the letter F . . . 24

6 The color of the background of the rows and columns **25**
- 6.1 Use of colortbl . . . 25
- 6.2 The tools of nicematrix in the \CodeBefore . . . 25
- 6.3 Color tools to be used inside the tabular . . . 30
- 6.4 The special color "nocolor" . . . 32

7 The command \RowStyle **32**

8 The width of the columns **33**
- 8.1 Basic tools . . . 33
- 8.2 The columns V of varwidth . . . 34
- 8.3 The columns X . . . 35

9 The exterior rows and columns **36**

10 Dotted leaders in the matrices **38**
- 10.1 The option xdots/nullify . . . 39
- 10.2 The commands \Hdotsfor and \Vdotsfor . . . 40
- 10.3 How to generate the dotted leaders transparently . . . 41
- 10.4 The labels of the dotted leaders . . . 42
- 10.5 Customisation of the dotted leaders . . . 42
- 10.6 The dotted leaders and the rules . . . 44
- 10.7 The commands \Hbrace and \Vbrace . . . 44

11 Delimiters in the preamble of the environment **45**

12 The \CodeAfter **46**

12.1 The command \line in the \CodeAfter . . . 47
12.2 The command \SubMatrix in the \CodeAfter (and the \CodeBefore) . . . 48
12.3 The commands \OverBrace and \UnderBrace in the \CodeAfter . . . 50
12.4 The command \TikzEveryCell . . . 51

13 Captions and notes in the tabulars **53**
13.1 Caption of a tabular . . . 53
13.2 The footnotes . . . 53
13.3 The notes of tabular . . . 53
13.4 Customisation of the tabular notes . . . 55

14 Other features **58**
14.1 Trees . . . 58
14.2 The key rounded-corners of the environments . . . 59
14.3 The command \ShowCellNames . . . 60
14.4 Use of the column type S of siunitx . . . 60
14.5 Default column type in {NiceMatrix} . . . 60
14.6 The command \rotate . . . 61
14.7 The key small . . . 62
14.8 \AutoNiceMatrix and the counters iRow and jCol . . . 62
14.9 The key light-syntax . . . 63
14.10 Color of the delimiters . . . 63
14.11 The environment {NiceArrayWithDelims} . . . 64
14.12 The command \OnlyMainNiceMatrix . . . 64

15 Explicit use of TikZ with nicematrix **64**
15.1 The nodes corresponding to the contents of the cells . . . 64
15.1.1 The key pgf-node-code . . . 66
15.1.2 The columns V of varwidth . . . 66
15.2 The medium nodes and the large nodes . . . 66
15.3 The nodes which indicate the position of the rules . . . 68
15.4 The nodes corresponding to the command \SubMatrix . . . 69

16 API for the developers **69**
16.1 Public sets of keys . . . 69
16.2 \CodeBefore and \CodeAfter . . . 70

17 Technical remarks **71**
17.1 Diagonal dotted leaders . . . 71
17.2 The empty cells . . . 72
17.3 The option exterior-arraycolsep . . . 72
17.4 Incompatibilities . . . 73
17.5 Compatibility with the Tagging Project of LaTeX . . . 73

18 Examples **74**
18.1 Notes in the tabulars . . . 74
18.2 Use of the key “tikz” of the command \Block . . . 75
18.3 Use with tcolorbox . . . 76
18.4 A sign chart . . . 76
18.5 A variation table . . . 77
18.6 Example of description of a table of a database . . . 77
18.7 Use of the key borders of the command \Block . . . 78
18.8 Leaders in the matrices . . . 78
18.9 How the illustrate the numbers of rows and columns in a matrix . . . 80
18.10 Stacks of matrices . . . 81
18.11 How to highlight cells of a matrix . . . 85
18.12 A triangular tabular . . . 87

Index **94**

1 The environments of this package

The package nicematrix defines the following new environments.

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

The environments {NiceArray}, {NiceTabular} and {NiceTabular*} are similar to the environments {array}, {tabular} and {tabular*} of the package array (which is loaded by nicematrix).

The environments {pNiceArray}, {bNiceArray}, etc. have no equivalent in array.

The environments {NiceMatrix}, {pNiceMatrix}, etc. are similar to the corresponding environments of amsmath (which is loaded by nicematrix): {matrix}, {pmatrix}, etc.

The environment {NiceTabularX} is similar to the environment {tabularx} from the eponymous package.[3].

The environment {NiceArrayWithDelims} is a generalisation of the environment {NiceArray} and its variants (cf. 14.11, p. 64).

It's recommended to use primarily the classical environments and to use the environments of nicematrix only when some feature provided by these environments is used (this will save memory and speed up the compilation).

By the way, when using an environment of nicematrix it's often possible to speed up the compilation by using the key no-cell-nodes (the compilation time may be reduced by a factor of 3 or 4). Indeed, numerous functionalities of nicematrix do not use the PGF/TikZ nodes created under the contents of the cells (cf. 15.1, p. 64) and the key no-cell-nodes avoids their creation which is expensive (the creation of those nodes can't be desactivated by nicematrix because nicematrix has no way to know whether those nodes are used directly by the final user in an environment {tikzpicture} after the environment of nicematrix).

All the environments of the package nicematrix accept, between square brackets, an optional list of *key=value* pairs. **There must be no space before the opening bracket ([) of this list of options.**

2 The vertical space between the rows

It's well known that some rows of the arrays created by default with LaTeX are, by default, too close to each other. Here is a classical example.

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

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



Inspired by the package cellspace which deals with that problem, the package nicematrix provides two keys cell-space-top-limit$^+$ and cell-space-bottom-limit$^+$ similar to the parameters of cellspace called \cellspacetoplimit and \cellspacebottomlimit.[4]

[3] In fact, it's possible to use directly the X columns in the environment {NiceTabular} (and the required width for the tabular is fixed by the key width): cf. 8.3, p. 35. Thus, it's possible to always avoid the environnement {NiceTabularX}

[4] One should remark that these parameters apply also to the columns of type S of siunitx whereas the package cellspace is not able to act on such columns of type S.

The initial values of these parameters is 0 pt.

There is also a key `cell-space-limits`$^+$ to set both parameters at once.

All these keys have a normal syntax but also an "additive" syntax. That means that it's possible to write, for example, `cell-space-limits = 2pt` but also `cell-space-limits += 1pt`.

```
$\begin{pNiceMatrix}[cell-space-limits = 1pt]
\frac12 & -\frac12 \\
\frac13 & \frac14 \\
\end{pNiceMatrix}$
```

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

It's also possible to change these parameters for only a few rows by using the command `\RowStyle` provided by nicematrix (cf. 7, p. 32).

3 The vertical position of the arrays

The package nicematrix provides a option `baseline` for the vertical position of the arrays. This option takes in as value an integer which is the number of the row on which the array will be aligned.

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

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

It's also possible to use the option `baseline` with one of the special values `t`, `c` or `b`. These letters may also be used absolutely like the option of the environments `{tabular}` and `{array}` of array. The initial value of `baseline` is `c`.

In the following example, we use the option `t` (equivalent to `baseline=t`) immediately after an `\item` of list. One should remark that the presence of a `\hline` at the beginning of the array doesn't prevent the alignment of the baseline with the baseline of the first row (with `{tabular}` or `{array}` of array, one must use `\firsthline`).

```
\begin{enumerate}
\item an 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. an item
2.

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

However, it's also possible to use the tools of booktabs[5]: `\toprule`, `\bottomrule`, `\midrule`, etc.

```
\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

It's also possible to use the key `baseline` to align a matrix on an horizontal rule (drawn by `\hline`). In this aim, one should give the value `line-`i where i is the number of the row *following* the horizontal rule.

```
\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 The blocks

4.1 General case

In the environments of nicematrix, it's possible to use the command `\Block` in order to place an element in the center of a rectangle of merged cells of the array.[6]
The command `\Block` must be used in the upper leftmost cell of the cells of the block with two mandatory arguments.

- The first argument is the size of the block with the syntax i`-`j where i is the number of rows of the block and j its number of columns.

  If this argument is empty, its default value is `1-1`. If the number of rows is not specified, or equal to `*`, the block extends until the last row (idem for the columns).

- The second argument is the content of the block. In `{NiceTabular}`, `{NiceTabular*}` and `{NiceTabularX}`, the content of the block is composed in text mode whereas, in the other environments, it is composed in math mode.

  Due to technical reasons of LaTeX, it's not possible to use `\color` at the beginnning of the content of the block (but it's possible to use `\textcolor`).[7]

[5] The extension booktabs is *not* loaded by nicematrix.
[6] The spaces after a command `\Block` are deleted.
[7] cf. `https://tex.stackexchange.com/questions/674788`

Here is an example of use of the command `\Block` in mathematical matrices.

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

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

One may wish to raise the size of the "A" placed in the block of the previous example. Since this element is composed in math mode, it's not possible to use directly a command like `\large`, `\Large` and `\LARGE`. That's why the command `\Block` provides an option between angle brackets to specify some TeX code which will be inserted before the beginning of the math mode.[8]

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

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

In fact, the command `\Block` accepts as first optional argument (between square brackets) a list of pairs *key=value*.
First, there are keys which are quick tools to control the appearance of the block.

- the key `fill` takes in as value a color and fills the block with that color;
- the key `opacity` sets the opacity of the filling color specified by `fill`;[9]
- the key `draw` strokes the frame of the block with the current color for the rules (however, it's possible to use another color by providing it as value of the key `draw`);
- the keys `hlines`, `vlines` and `hvlines` draw all the corresponding rules in the block;[10]
- the key `rules/width` is the width of the rules (is relevant only when one of the keys `draw`, `hvlines`, `vlines` and `hlines` is used);
- the key `rules/width` is the width of the rules (is relevant only when one of the keys `hvlines`, `vlines` and `hlines` is used);
- the key `rounded-corners` requires rounded corners (for the frame drawn by `draw` and the shape drawn by `fill`) with a radius equal to the value of that key (the default value is 4 pt, which is the initial value of the *rounded corners* of TikZ)[11] ;

Sometimes, these tools are not sufficient to control the appearance of the block. The following keys are more powerful but also more difficult to use. Moreover, the require the loading of TikZ by the user (with `\usepackage{tikz}`). By default, nicematrix does not load TikZ but only PGF, which is a sublayer of TikZ.

[8] This argument between angular brackets may also be used to insert a command of font such as `\bfseries` when the command `\\` is used in the content of the block. It's also possible to put in that optional argument the command `\rotate` provided by nicematrix (cf. 14.6, p. 61).

[9] Caution: that feature creates instructions of transparency in the PDF and some readers of PDF don't support such instructions. The opacity is applied by using `\pgfsetfillopacity`.

[10] However, the rules are not drawn in the sub-blocks of the block, as always with nicematrix: the rules are not drawn in the blocks, except when they have the key `transparent` (cf. 5 p. 14).

[11] There is also a key `rounded-corners` available for an environment of nicematrix (cf. 14.2, p. 59).

- The key `borders` provides the ability to draw only some borders of the blocks; the value of that key is a (comma-separated) list of elements covered by `left`, `right`, `top`, `bottom` and `all`; it's possible, in fact, in the list which is the value of the key `borders`, to add an entry of the form `tikz={`*list*`}` where *list* is a list of couples *key=value* of TikZ specifying the graphical characteristics of the lines that will be drawn (cf. example 18.7, p. 78). When the key `borders` is used without value, it is equivalent to the key `draw` (without value).

- When the key `tikz` is used, the TikZ path corresponding of the rectangle which delimits the block is executed with TikZ[12] by using as options the value of that key `tikz` (which must be a list of keys allowed for a TikZ path).

  In fact, in the list of the keys provided by the user as value of `tikz`, it's possible to put a key `offset`. That key is not provided by TikZ but by nicematrix. It will narrow the rectangular frame corresponding to the block by a margin (horizontally and vertically) equal to the value (of that key `offset`). That new frame, a bit narrower, will be executed by TikZ with options which are the other keys in the list of keys provided as value to the key `tikz` of `\Block`.

  New 7.11 : Two more precise keys, `xoffset` and `yoffset`, are provided.

There is also some technical keys:

- the key `name` provides a name to the rectangular TikZ node corresponding to the block; it's possible to use that name with TikZ in the `\CodeAfter` of the environment (cf. 12, p. 46);

- the key `respect-arraystretch` prevents the setting of `\arraystretch` to 1 at the beginning of the block (which is the behaviour by default) ;

- By default, the rules are not drawn in the blocks (see the section about the rules: section 5, p. 14). However, if the key `transparent` is used, the rules are drawn. For an example, see section 18.2, p. 75. Caution: that key does not imply that the content of the block will be transparent!

- **New 7.12**

  With the key `empty`, all the cells of the block are declared as "empty", which has impact on several functionalities of nicematrix, for example the key `corners` for the computation of the "empty corners" (cf. 5.3.3, p. 17).

There is also keys for the horizontal and vertical positions of the content of the block: cf. 4.5, p. 10.

One must remark that, by default, the commands `\Blocks` don't create space. There is exception only for the blocks mono-column and the blocks mono-row under some conditions as explained just below.

In the following example, we have had to enlarge by hand the columns 2 and 3 (with the construction `w{c}{...}` of array).

```
\begin{NiceTabular}{cw{c}{2cm}w{c}{3cm}c}
  rose & tulip & daisy & dahlia \\
  violet
    & \Block[C,draw=red,fill=[RGB]{204,204,255},rounded-corners]{2-2}
            {\LARGE Some beautiful flowers} & & marigold \\
  iris & & & lis \\
  arum & periwinkle & forget-me-not &  hyacinth
\end{NiceTabular}
```



[12] TikZ should be loaded (by default, nicematrix only loads PGF) and, if it's not, an error will be raised.

4.2 The mono-column blocks

The mono-column blocks have a special behaviour.

- The natural width of the contents of these blocks is taken into account for the width of the current column.

  In the columns with a fixed width (columns `w{...}{...}`, `W{...}{...}`, `p{...}`, `b{...}`, `m{...}`, `V` and `X`), the content of the block is formatted as a paragraph of that width.

- The specification of the horizontal position provided by the type of column (`c`, `r` or `l`) is taken into account for the blocks. For a block in a column ot type `p{...}`, `b{...}`, `m{...}`, `V{...}` ou `X`, the alignment `c` will be used by default. However, those types of columns may have an optional argument for the horizontal alignment (eg: `p[l]{...}`) and, in that case, that type of alignement is passed to the block.

  Of course, the `\Block` may also have its own specification of alignment: cf. 4.5, p. 10.

- The specifications of font specified for the column by a construction `>{...}` in the preamble of the array are taken into account for the mono-column blocks of that column (this behaviour is probably expected).

```
\begin{NiceTabular}{@{}>{\bfseries}lr@{}} \hline
  \Block{2-1}{John}    & 12 \\
                       & 13 \\ \hline
  Steph                &  8 \\ \hline
  \Block{3-1}{Sarah}   & 18 \\
                       & 17 \\
                       & 15 \\ \hline
  Ashley               & 20 \\ \hline
  Henry                & 14 \\ \hline
  \Block{2-1}{Madison} & 15 \\
                       & 19 \\ \hline
\end{NiceTabular}
```

John	12
	13
Steph	8
Sarah	18
	17
	15
Ashley	20
Henry	14
Madison	15
	19



4.3 The mono-row blocks

For the mono-row blocks, the natural height and depth are taken into account for the height and depth of the current row (as does a standard `\multicolumn` of LaTeX), except when an option of vertical position has been used for the block (one of the keys `t`, `b`, `m`, `T` and `B` described in the part 4.6, p. 11).

4.4 The mono-cell blocks

A mono-cell block inherits all the properties of the mono-row blocks and mono-column blocks.

At first sight, one may think that there is no point using a mono-cell block. However, there are some good reasons to use such a block.

- It's possible to use the command `\\` in a (mono-cell) block.
- It's possible to split the cell in several sub-cells with `&` when the key `&-in-blocks` is in force (cf. 4.7, p. 12).
- It's possible to use the option of horizontal alignment of the block in derogation of the type of column given in the preamble of the array.
- It's possible do draw a frame around the cell with the key `draw` of the command `\Block` and to fill the background with rounded corners with the keys `fill` and `rounded-corners`.[13]

[13] If one simply wishes to color the background of a unique cell, there is no point using the command `\Block`: it's possible to use the command `\cellcolor`.

- It's possible to draw one or several borders of the cell with the key `borders`.

4.5 Horizontal position of the content of the block

The command `\Block` accepts the keys `l`, `c` and `r` for the horizontal position of its content.

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

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

By default, the horizontal position of the content of a block is computed by using the positions of the *contents* of the columns implied in that block. That's why, in the following example, the header "First group" is correctly centered despite the instruction `!{\qquad}` in the preamble which has been used to increase the space between the columns (this is not the behaviour of `\multicolumn`).

```
\begin{NiceTabular}{@{}c!{\qquad}ccc!{\qquad}ccc@{}}
\toprule
  Rank & \Block{1-3}{First group} & & & \Block{1-3}{Second group} \\
       & 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}
```

Rank	First group			Second group		
	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

In order to have an horizontal positioning of the content of the block computed with the limits of the columns of the LaTeX array (and not with the contents of those columns), one may use the key `L`, `R` and `C` of the command `\Block`.[14]

Here is the same example with the key `C` for the first block.

```
\begin{NiceTabular}{@{}c!{\qquad}ccc!{\qquad}ccc@{}}
\toprule
  Rank & \Block[C]{1-3}{First group} & & & \Block{1-3}{Second group} \\
       & 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}
```

[14] Remark: The keys `L`, `C` and `R` require less computations than the keys `l`, `c` and `r`. If you are concernend about efficiency, you should use by default `\Block[C]`.

Rank	First group			Second group		
	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

The command `\Block` supports also the keys `p` and `j`. With the key `p`, the content of the block is formatted like a paragraph (as in a column of type `p`). That key may be used in conjunction with a key `l`, `c` or `r`, and, in that case, the paragraph is formatted with `\raggedright`, `\centering` or `\raggedleft` (or `\RaggedRight`, `\Centering` and `\RaggedLeft` when `ragged2e` is loaded). With the key `j` (which inforces the key `p`), the paragraph is justified.

It's possible to put an environment `{itemize}` or `{enumerate}` in a block which uses the key `p` or `j` (in the other case, you will have a LaTeX error: `Not allowed in LR mode`). For the following example, we have loaded the package enumitem (for the key `left` of the environment `{itemize}`).

```
\usepackage{enumitem} % in the preamble

\begin{NiceTabular}[hvlines]{ccc}
  one & two two & three three \\
  one &
  \Block[p]{*-2}{%
  \begin{itemize}[left=0pt]
  \item one two three four five
  \item two
  \item three
  \end{itemize}
  } \\
  one & \\
  one & \\
  one & \\
  one & \\
  one & \\
  one & \\
\end{NiceTabular}
```

one	two two	three three
one		
one	• one two three four five	
one		
one	• two	
one		
one	• three	
one		



4.6 Vertical position of the content of the block

For the vertical position, the command `\Blocks` supports the keys `m`, `t`, `b`, `T` and `B`.

- With the key `m`[15], the content of the block is vertically centered.
- With the key `t`, the baseline of the content of the block is aligned with the baseline of the first row concerned by the block.
- with the key `b`, the baseline of the last row of the content of the block (we recall that the content of a block may contain several lines of text separated by `\\`) is aligned with the baseline of the last of the rows of the array involved in the block.
- With the key `T`, the content of the block is set upwards.

  No vertical margin is added. However, the contents of the block is (always) composed by nicematrix in a `{minipage}`, a `{tabular}` or an `{array}` and, hence, there will still remain a margin (in most cases). If needed, it's always possible to add a `\strut`...

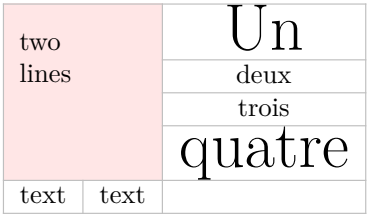
[15] That key has an alias: `v-center`.

- With the key `B`, the content of the block is set downwards.

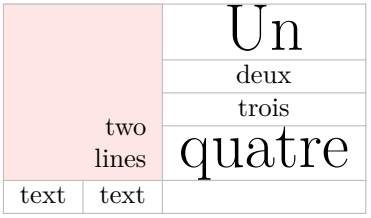
When no key is given, the key `m` applies (except in the mono-row blocks).

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

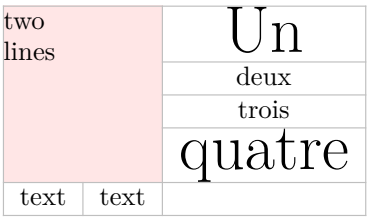
\begin{NiceTabular}{ccc}
  \Block[fill=red!10,t,l]{4-2}{two\\lines}
   & & \Huge Un\\
   & & deux  \\
   & & trois  \\
   & & \Huge quatre  \\
  text & text \\
\end{NiceTabular}
```



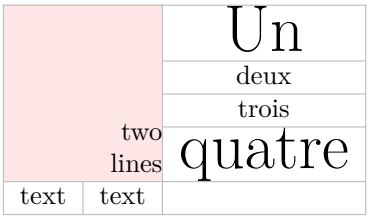

```
\begin{NiceTabular}{ccc}
\Block[fill=red!10,b,r]{4-2}{two\\lines}
   & & \Huge Un\\
   & & deux  \\
   & & trois  \\
   & & \Huge quatre  \\
  text & text \\
\end{NiceTabular}
```




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




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



4.7 \\ and & in the blocks

The extension nicematrix provides the ability to use `\\` and `&` directly in the content of a block (in order to format its contents) but there is some rectrictions.

- One must not use both `\\` and `&` in the same block.
- For `\\`, there is no other restriction. It's possible to use `\\` in a block to format a text on several lines.
- In order to use `&`, the key `ampersand-in-blocks` (alias: `&-in-blocks`) must been activated[16]. Then, the block is divided in sub-blocks as illustrated below. Be careful: when `ampersand-in-blocks` is in force, the (main) argument of the command `\Block` is syntactically divided

[16] Otherwise, the use of `&` in the command `\Block` will raise a TeX error :
`! Extra alignment tab has been changed to \cr.`

into sub-blocks by splitting on the ampersands `&`, the ampersands between curly braced are protected but not those in an environment.[17]

With the ampersand `&`, it's possible to divide horizontally a block in sub-blocks of *the same size*.

```
\begin{NiceTabular}{ll}%
   [hvlines,ampersand-in-blocks]
    & the five first naturels numbers \\
  3 & \Block{}{one & two & three} \\
  4 & \Block{}{one& two & three & four} \\
  5 & \Block{}{one & two & three & four & \
+ five} \\
\end{NiceTabular}
```

	the five first naturels numbers				
3	one	two	three		
4	one	two	three	four	
5	one	two	three	four	five

As can be seen, the blocks (which was are in fact mono-cell blocks) are divided into sub-blocks of the *same size*. However, maybe the following code would be prefered.

```
\begin{NiceTabular}{lccccc}%
   [hvlines,ampersand-in-blocks]
    & \Block{1-5}{the five first
                  natural numbers} \\
  3 & \Block{1-5}{one & two & three} \\
  4 & \Block{1-5}{one& two &
        \cellcolor{red!15} three & four} \\
  5 & one & two & three & four & five \\
\end{NiceTabular}
```

	the five first natural numbers				
3	one	two	three		
4	one	two	three	four	
5	one	two	three	four	five

In this code, we have blocks of size `1-5` which are divided into three or four sub-blocks.

We have illustrated the fact that it's possible to color the background of a subcell of the block by using the key `\cellcolor`.

4.8 The key create-blocks-in-col

The key `create-blocks-in-col` takes in as argument a list of numbers of columns and, for each of those columns, creates within that column a block for each non-empty cell (that block will enclose that non-empty cell and all the following empty cells in the column).
These blocks will be, as all the blocks, taken into account for the rules (drawn by `vlines`, `hlines`, etc.: cf. 5, p. 14) and the commands that color the background of the cells (cf. 6.2, p. 25). That the point using the key `create-blocks-in-col`.

```
\begin{NiceTabular}{lr}[hvlines,create-blocks-in-col=1]
\rowcolors{blue!10}{}[respect-blocks]
  John      & 12 \\
            & 13 \\
  Steph     &  8 \\
  Sarach    & 18 \\
            & 17 \\
            & 15 \\
  Ashley    & 20 \\
  Henry     & 14 \\
  Madison   & 15 \\
            & 19
\end{NiceTabular}
```

John	12
	13
Steph	8
Sarach	18
	17
	15
Ashley	20
Henry	14
Madison	15
	19

[17] It's not possible to write `\Block[ampersand-in-blocks]{}{\begin{array}{cc}1&2\end{array}}`. Of course, it's possible without the key `ampersand-in-blocks`.

5 The rules

The usual techniques for the rules may be used in the environments of nicematrix (except `\vline`). However, there is some small differences with the classical environments.

5.1 Some differences with the classical environments

5.1.1 The vertical rules

In the environments of nicematrix, the vertical rules specified by `|` in the preambles of the environments are never broken, even by an incomplete row or by a double horizontal rule specified by `\hline\hline` (there is no need to use the package hhline[18]).

```
\begin{NiceTabular}{|c|c|} \hline
  First & Second \\ \hline\hline
  Peter \\ \hline
  Mary & George\\ \hline
\end{NiceTabular}
```

First	Second
Peter	
Mary	George



However, the vertical rules are not drawn in the blocks (created by `\Block`: cf. 4, p. 6) nor in the corners (created by the key `corner`: cf. 5.3.3, p. 17) nor in the potential exterior rows (created by the keys `first-row` and `last-row`: cf. 9, p. 36).

If you use `\multicolumn` there is no need to add the potential `|` or `||` at the end of the little preamble (but, in the environments of nicematrix, it's often more practicable to use the command `\Block`).

If you use booktabs (which provides `\toprule`, `\midrule`, `\bottomrule`, etc.) and if you actually want to add vertical rules (which is not in the spirit of booktabs), you should notice that the vertical rules drawn by nicematrix are compatible with booktabs. Remark that nicematrix does *not* load 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



However, it's still possible to define a specifier (named, for instance, `I`) to draw vertical rules with the standard behaviour of array. In this aim, we use the standard command `\newcolumntype` of array.

```
\newcolumntype{I}{!{\vrule}}
\begin{NiceTabular}{IcIcI} \hline
  First & Second \\ \hline\hline
  Peter & \\ \hline
  Mary  & George\\ \hline
Premier & Deuxième \\ \hline\hline
Paul  & \\ \hline
Marie & Pauline \\ \hline
\end{NiceTabular}
```

First	Second
Peter	
Mary	George
Premier	Deuxième
Paul	
Marie	Pauline



Of course, in this case, there is no point using `{NiceTabular}` and it's better to use the standard environment `{tabular}` but maybe we will have to use other functionalities provided by nicematrix. One should also consider the command `\FalseRow` (cf. 5.3.8, p. 24) for the tables with horizontal double rules.

[18] But hhline may be used with nicematrix: see the example 18.6, p. 77.

5.1.2 The command \cline

The horizontal and vertical rules drawn by \hline and the specifier "|" make the array larger or wider by a quantity equal to the width of the rule (with array and also with nicematrix).

For historical reasons, this is not the case with the command \cline, as shown by the following example.

```
\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

In the environments of nicematrix, this situation is corrected (it's still possible to go to the standard behaviour of \cline with the key `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

In the environments of nicematrix, an instruction \cline{*i*} is equivalent to \cline{*i*-*i*}.

5.2 The thickness and the color of the rules

The environments of nicematrix provide a key `rules/width` to set the width (in fact the thickness) of the rules in the current environment. In fact, this key merely sets the value of the standard LaTeX length \arrayrulewidth.

It's well known that colortbl provides the command \arrayrulecolor in order to specify the color of the rules.

With nicematrix, it's possible to specify the color of the rules even when colortbl is not loaded. For sake of compatibility, the command is also named \arrayrulecolor. However, nicematrix also provides a key `rules/color`, available in \NiceMatrixOptions or in an individual environment, to fix the color of the rules. This key sets the value locally (whereas \arrayrulecolor acts globally!) and should be preferred.

```
\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

In fact, in that example, instead of \hline, it would have a better choice to use \Hline, provided by nicematrix and described just below, because it ensures a better output in the PDF viewers at the low levels of zoom.

5.3 The tools of nicematrix for the rules

Here are the tools provided by nicematrix for the rules.

- the keys `hlines`, `vlines`, `hvlines` and `hvlines-except-borders`;
- the specifier "|" in the preamble (for the environments with preamble);
- the command \Hline.

All these tools don't draw the rules in the blocks nor in the empty corners (when the key `corners` is used), nor in the exterior rows and columns.

- These blocks are:
  - the blocks created by the command `\Block`[19] (cf. 4, p. 6);
  - the blocks createad by the key `create-blocks-in-col`: (cf. 4.8, p. 13);
  - the blocks implicitly delimited by the dotted leaders created by `\Cdots`, `\Vdots`, etc. (cf. 10, p. 38).
- The corners are created by the key `corners` explained below (cf. 5.3.3, p. 17).
- For the exterior rows and columns, cf. 9, p. 36.

In particular, this remark exhibits a first difference between the standard command `\hline` and the command `\Hline` provided by nicematrix.
Moreover, the key `\Hline` takes in an optional argument (between square brackets) which is a list of *key=value* pairs. For the description of those keys, see `custom-line`, in the section 5.3.5, p. 19.[20]

As well as the command `\Hline`, the specifier "`|`" supports an optional argument between square brackets for the characteristics of the rule.

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

a	b
c	d

5.3.1 The keys hlines and vlines

The keys `hlines` and `vlines` draw horizontal and vertical rules.
If no value is provided, all the rules are drawn.

When a value is provided, it should be a comma-separated list of numbers (corresponding to the rules which will be drawn).

- It's possible to put some intervals of integers with the syntax *i*`-`*j*
- It's possible to put negative numbers (which are counted from the end).

In fact, for the environments with delimiters (such as `{pNiceMatrix}` or `{bNiceArray}`), the key `vlines` don't draw the exterior rules (this is certainly the expected behaviour).

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

$$\left(\begin{array}{c|c|c|c|c|c} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{array}\right)$$

When the key `hlines` is in force, it's still possible to use `\Hline\Hline` to put a double horizontal rule. As well, it's possible to put `||` in the preamble (of an environment with preamble) to put a double vertical rule, even when the key `vlines` is in force.

[19] And also the command `\multicolumn` but it's recommended to use instead `\Block` in the environments of nicematrix (by the way, the command `\multicolumn` has the feature, unlike `\Block`, that it enlarges the last column if necessary).
[20] Technical remark: If the user wants to write a command over the command `\Hline`, it shall be ensured that this new command is expandable in the TeX sens (by using, for instance, `\NewExpandableDocumentCommand` of LaTeX3, `\newcommand` of LaTeX or `\def` of TeX). Example: `\NewExpandableDocumentCommand{}{}{\Hline[color=red]}`

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

	a	b	c	d	e
x	0	0	0	0	0
y	0	0	0	0	0
z	0	0	0	0	0



5.3.2 The keys hvlines, hlines-except-borders and hvlines-except-borders

The key `hvlines` (no value) is the conjunction of the keys `hlines` and `vlines`.

```
\begin{NiceTabular}{cccc}[hvlines,rules={color=blue,width=1pt}]
  rose      & tulip & sunflower & lily \\
  violet    & \Block[C,draw=red]{2-2}{\LARGE flowers} & & orchid \\
  carnation & & & lys \\
  daisy     & peony & jasmine & magnolia
\end{NiceTabular}
```



rose	tulip	sunflower	lily
violet	flowers		orchid
carnation			lys
daisy	peony	jasmine	magnolia

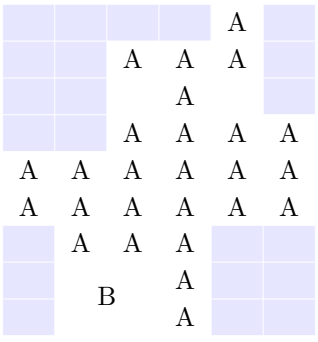
It's worth noting that, when the key `rounded-corners` is used for the environment `{NiceTabular}`, the key `hvlines` draws rounded corners for the exterior frame of the tabular (cf. 14.2, p. 59).

The key `hlines-except-borders` is similar to the key `hlines` but does not draw the rules on the horizontal borders of the array. The key `hvlines-except-borders` is similar to the key `hvlines` but does not draw the rules on the horizontal and vertical borders of the array. For an example of use of that key, see the part "Use with tcolorbox" (18.3, p. 76).

5.3.3 The (empty) corners

The four `corners` of an array will be designed by `NW`, `SW`, `NE` and `SE` (*north west*, *south west*, *north east* and *south east*).
For each of these corners, we will call *empty corner* (or simply *corner*) the reunion of all the empty rectangles starting from the cell actually in the corner of the array.[21]
However, it's possible, for a cell without content, to require nicematrix to consider that cell as not empty with the command `\NotEmpty`.

In the example on the right (where B is in the center of a block of size 2×2), we have colored in blue the four (empty) corners of the array.

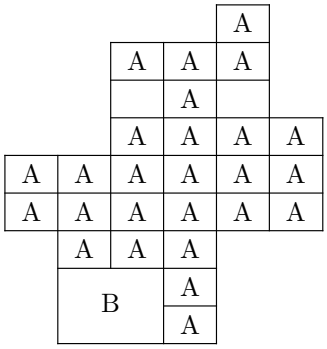


When the key `corners`[22] is used, nicematrix computes the four (empty) corners and these corners will be taken into account by the tools for drawing the rules (the rules won't be drawn in the corners).

[21] For sake of completeness, we should also say that a cell contained in a block (even an empty cell) is not taken into account for the determination of the corners. That behaviour is natural. The precise definition of a "non-empty cell" is given below (cf. 17.2, p. 72).

[22] The key `corners` that we describe now has no direct link with the key `rounded-corners` described in the part 14.2, p. 59.

```
\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[C]{2-2}{B} & & A \\
    &   &   & A \\
\end{NiceTabular}
```



It's also possible to provide to the key `corners` a (comma-separated) list of corners (designed by `NW`, `SW`, `NE` and `SE`).

```
\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

▷ The corners are also taken into account by the tools provided by nicematrix to color cells, rows and columns. These tools don't color the cells which are in the corners (cf. 6.2, p. 25). The command `\TikzEveryCell` available in the `\CodeBefore` and the `\CodeAfter` (cf. 12.4, p. 51) takes also into account the empty corners.

New 7.10

A generalization of the «corners» has been made with the new values `N`, `S`, `E` and `W` for the «corners».

```
\begin{NiceTabular}{cccc}[hvlines,corners=E]
1 & 2 \\
1 & 2 & 3 \\
1 & 3 \\
1 \\
1 & 2 & 3 & 4 \\
\end{NiceTabular}
```

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

5.3.4 The command \diagbox

The command `\diagbox` (inspired by the package diagbox), allows, when it is used in a cell, to slash that cell diagonally downwards.

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

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

It's possible to use the command `\diagbox` in a `\Block`.

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

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

But it's also possible to use `\diagbox` for the diagonal rule only (and the labels are put but the usual way, that is to say in the cells of the tabular).

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

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

Nevertheless, it's always possible to draw whatever rule we wish with TikZ in the `\CodeAfter` (or the `\CodeBefore`) by using the PGF/TikZ nodes created by nicematrix: cf. 15, p. 64.

5.3.5 Commands for customized rules

It's also possible to define commands and letters for customized rules with the key `custom-line` available in `\NiceMatrixOptions` and in the options of individual environments. That key takes in as argument a list of *key=value* pairs. First, there is three keys to define the tools which will be used to use that new type of rule.

- the key `command` is the name (*with* or *without* the backslash) of a command that will be created by nicematrix and that will be available for the final user in order to draw horizontal rules (similarly to `\hline`);
- the key `ccommand` is the name (*with* or *without* the backslash) of a command that will be created by nicematrix and that will be available for the final user to order to draw partial horizontal rules (similarly to `\cline`, hence the name `ccommand`): the argument of that command will be list of intervals of columns specified by the syntax i or i-j.[23]
- the key `letter` takes in as argument a letter[24] that the user will use in the preamble of an environment with preamble (such as `{NiceTabular}` in order to specify a vertical rule.

We will now speak of the keys which describe the rule itself. Those keys may also be used in the (optional) argument of an individual command `\Hline` or in the (optional) argument of a specifier "`|`" in the preamble of an environment. They are also used for the key `default-line`.

There are four possibilities:

- the basic rules, which may be drawn by using (for example) colortbl: these are simple or multiple rules, with color and a color between the rules;
- the dashed rules (key `dashed`);
- the dotted rules (key `dotted`);
- all the rules which may be draw with TikZ (key `tikz`).

[23] It's recommended to use such commands only once in a row because each use will create space between the rows corresponding to the total width of the rule. Likewise, it's possible to draw several rules with only one use of the command.

[24] The following letters are forbidden: `lcrpmbFVX|()[]!@<>`

We will now describe these four kinds of rules.

- *First possibility*

  It's possible to specify composite rules, with a color and a color for the inter-rule space (as possible with colortbl for instance).

  - the key `multiplicity` is the number of consecutive rules that will be drawn: for instance, a value of 2 will create double rules such those created by `\hline\hline` or `||` in the preamble of an environment;
  - the key `color` sets the color of the rules ;
  - the key `sep-color` sets the color between two successive rules (should be used only in conjunction with `multiplicity`). The name of that key is inspired by the command `\doublerulesepcolor` of colortbl.

  That system may be used, in particular, for the definition of commands and letters to draw rules with a specific color (and those rules will respect the blocks and corners as do all the rules of nicematrix).

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



	dimensions		
	L	l	h
Product A	3	1	2
Product B	1	3	4
Product C	5	4	1

  The key `sep-color` with the value `white` may also be used in case of an horizontal double-rule on the top of a colored cell (unlike the case with `{tabular}` and colortbl, the color applies by default between the rules).

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

one	two	three
four	five	six

```
\begin{NiceTabular}{ccc}
  one & two & three \\
\DoubleRule
  four & \cellcolor{yellow} five & six \\
\end{NiceTabular}
```



  However, for double rules in conjunction with colored backgrounds, it's probably more advisable to use the "false rows" provided by the command `\FalseRow` (dans the letter `F` for the false columns): cf. 5.3.8, p. 24.

- *Second possibility* : the key `dashed`

  The extension nicematrix provides the commands `\hdashedline` and `\cdashedline` and the letter « ; » for dashed rules. They are available when TikZ is loaded because it's merely a line of TikZ with the style `dash pattern = on 4 pt off 2pt`. In fact, when the TikZ library decorations is loaded (with `\usetikzlibrary{decorations}`), the key `dash expand off` is added for a better output.

  Thus, it's possible to use the commands `\hdashedline` and `\cdashedline` to draw horizontal dashed rules.

```
\usepackage{tikz}           % in the preamble
\usetikzlibrary{decorations} % in the preamble

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

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

  In the environments with an explicit preamble (like `{NiceTabular}`, `{NiceArray}`, etc.), it's possible to draw a vertical dashed rule with the specifier "`;`".

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

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

  If you wish a more important reservation of space for those rules, you should define, with `custom-line`, some new commands by using the key `dashed` in conjonction with the key `total-width`.

```
\NiceMatrixOptions
  {
    custom-line =
     {
       command = \myhdashedline ,
       ccommand = \mycdashedline ,
       dashed,
       total-width = 10 pt
     }
  }

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

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

- *Third possibility* : the key `dotted`

  The extension nicematrix provides the commands `\hdottedline` and `\cdottedline` and the letter « : » for dotted lines. For these lines, nicematrix does not use TikZ but an internal system. In fact, it's the same as the system used by the commands `\Cdots`, `\Vdots`, etc. (cf. 10, p. 38).

Thus, it's possible to use the commands `\hdottedline` and `\cdottedline` to draw horizontal dotted rules.

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

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

In the environments with an explicit preamble (like `{NiceTabular}`, `{NiceArray}`, etc.), it's possible to draw a vertical dotted rule with the specifier "`:`".

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

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

Like with the key `dashed` described previously, it's possible to use this key in conjunction with the key `total-width`.

- *Fourth possibility* : the key `tikz`

  It's possible to use the key `tikz` (if TikZ is loaded). In that case, the rule is drawn directly with TikZ by using as parameters the value of the key `tikz` which must be a list of *key=value* pairs which may be applied to a TikZ path.

  By default, no space is reserved for the rule that will be drawn with TikZ. It is possible to specify a reservation (horizontal for a vertical rule and vertical for an horizontal one) with the key `total-width`. That value of that key, is, in some ways, the width of the rule that will be drawn (nicematrix does not compute that width from the characteristics of the rule specified in `tikz`).

```
\usepackage{tikz}                       % in the preamble
\usetikzlibrary{decorations.pathmorphing} % in the preamble

\NiceMatrixOptions
  {
    custom-line =
     {
       letter = I ,
       tikz = { decorate, decoration = zigzag } ,
       total-width = 6 pt
     }
  }

\begin{NiceTabular}{cIcIc}
  one & two & three \\
  four & five & six \\
  seven & eight & nine
\end{NiceTabular}
```

one	two	three
four	five	six
seven	eight	nine

As we have said, the previous keys may be used in the optional argument of an individual command `\Hline` and in the optional argument of a specifier "`|`" in the preamble of an environment (for instance `{NiceTabular}`).
However, in that case, it's also possible to use two other keys, `start` and `end`, which specify the extremities of the rule.

```
\begin{NiceTabular}{cc|[color=blue,start=2]ccc}
  one & two & three & four
\Hline[start=2,end=3]
  five & six & seven & eight \\
  nine & ten & eleven & twelve
\end{NiceTabular}
```

one	two	three	four
five	six	seven	eight
nine	ten	eleven	twelve

5.3.6 The key fix-vertex

New 7.9

Assume that we have defined a command `\RandomLine` for a line which zigzags randomly:

```
\usepackage{tikz}                          % in the preamble
\usetikzlibrary{decorations.pathmorphing} % in the preamble

\NiceMatrixOptions
  {
    custom-line =
     {
       command = \RandomLine ,
       tikz =
         {
           decorate ,
           decoration = { random steps, segment length = 5pt, amplitude = 3pt }
         }
     }
  }
```

Let's consider the following tabular, with `\RandomLine` used at the end:

```
\begin{NiceTabular}{ccc}[vlines]
\Hline
  Name & First name & Age \\
\Hline
  Dubourg & Claude & 88 \\
  Challa & Remy & 72 \\
  Ducar & Estelle & 75 \\
  Migaud & Sophie & 18 \\[1mm]
\RandomLine
\end{NiceTabular}
```

Name	First name	Age
Dubourg	Claude	88
Challa	Remy	72
Ducar	Estelle	75
Migaud	Sophie	18

The internal vertical rules won't always stop exactly at the bottom line since that line goes randomly from the bottom left corner to the bottom right one.
In order to address this problem, nicematrix provides the key `rules/fix-vertex`. When that key is in force, the rules are divided in segments between the vertices and those segments are drawn independently of each other.

```
\begin{NiceTabular}{ccc}[vlines,rules/fix-vertex]
\Hline
  Name & First name & Age \\
\Hline
  Dubourg & Claude & 88 \\
  Challa & Remy & 72 \\
  Ducar & Estelle & 75 \\
  Migaud & Sophie & 18 \\[1mm]
\RandomLine
\end{NiceTabular}
```

Name	First name	Age
Dubourg	Claude	88
Challa	Remy	72
Ducar	Estelle	75
Migaud	Sophie	18

5.3.7 The key default-line

New 7.9

The package nicematrix provides the key `default-line` which sets the style of all the rules that will be drawn by the command `\Hline`, the specifier `|` in the preambles of the environments and the keys `hvlines`, `vlines`, `hlines`, etc. The different ways for the specification of the rules are the sames as those for the key `custom-line` (cf. 5.3.5, p. 19).

```
\usepackage{tikz}                          % in the preamble
\usetikzlibrary{decorations.pathmorphing} % in the preamble

\NiceMatrixOptions
  {
    default-line =
      {
        tikz =
          {
            decorate ,
            decoration = { random steps, segment length = 3pt, amplitude = 1pt }
          }
      }
  }

\renewcommand{\arraystretch}{1.3}
\begin{NiceTabular}{ccc}[hvlines,rules/fix-vertex]
  one   & two  & three \\
  four  & five & six \\
  seven & \Block{1-2}{eight} \\
\end{NiceTabular}
```

one	two	three
four	five	six
seven	eight	

5.3.8 The command \FalseRow and the letter F

New 7.11

For double rules perfectly compatible with the colored backgrounds, nicematrix provides a command `\FalseRow` and a letter `F`.

The command `\FalseRow` must be used after a command `\\` of end of row. It creates a new empty row with a height equal to `width-of-false`$^{+}$ (the initial value of that parameter is 2 pt, which is the initial value of `\doublerulesep`). The rules and the colored backgrounds won't be drawn in that row.

The letter `F` is available in the environments with preambles (`{NiceTabular}`, `{NiceArray}`, etc.). It creates a column with a width equal to the parameter `width-of-false`. The rules and the colored backgrounds won't bex drawn in that column. Caution: that column is a actual column of the tabular and two ampersands (`&&`) must be used in the rows that cross that column.

```
\begin{NiceTabular}{cFcc}[hvlines]
\CodeBefore
  \rowcolor[opacity=0.5]{red!50}{3}
  \columncolor[opacity=0.5]{blue!50}{3}
\Body
\diagbox{x}{y} && a & b \\
\FalseRow
a              && b & a \\
b              && a & b
\end{NiceTabular}
```

x \ y	a	b
a	b	a
b	a	b

New 7.12

A key `color-of-false` is provided for the color of the backgrounds of those false rows and false columns.

6 The color of the background of the rows and columns

6.1 Use of colortbl

We recall that the package colortbl can be loaded directly with `\usepackage{colortbl}` or by loading xcolor with the key `table`: `\usepackage[table]{xcolor}`.

However, there is two drawbacks:

- The package colortbl patches array, leading to some incompatibilities (for instance with the command `\hdotsfor`).
- The package colortbl constructs the array row by row, alterning colored rectangles, rules and contents of the cells. The resulting PDF is difficult to interpret by some PDF viewers and may lead to artefacts on the screen.
  - Some rules seem to disappear. This is because many PDF viewers give priority to graphical element drawn posteriorly (which is in the spirit of the "painting model" of PostScript and PDF). Concerning this problem, MuPDF (which is used, for instance, by SumatraPDF) gives better results than Adobe Reader.
  - A thin white line may appear between two cells of the same color. This phenomenon occurs when each cell is colored with its own instruction `fill` (the PostScript operator `fill` noted `f` in PDF). This is the case with colortbl: each cell is colored on its own, even when `\columncolor` or `\rowcolor` is used.

    As for this phenomenon, Adobe Reader gives better results than MuPDF.

  The package nicematrix provides tools to avoid both problems.

6.2 The tools of nicematrix in the `\CodeBefore`

The package nicematrix provides some tools to draw the colored panels first, and, then, the content of the cells and the rules. This strategy is more conform to the "painting model" of the formats PostScript and PDF and is more suitable for the PDF viewers. However, it requires several compilations.[25]

The extension nicematrix provides a key `code-before` for some code that will be executed before the drawing of the tabular.

An alternative syntax is provided: it's possible to put the content of that `code-before` between the keywords `\CodeBefore` and `\Body` at the beginning of the environment.

```
\begin{pNiceArray}{preamble}
\CodeBefore [options]
  instructions of the code-before
\Body
  contents of the environment
\end{pNiceArray}
```

The optional argument between square brackets is a list of *key=value* pairs which will be presented progressively in this documentation.[26]

New commands are available in that `\CodeBefore`: `\cellcolor`, `\rectanglecolor`, `\rowcolor`, `\columncolor`, `\rowcolors`, `\rowlistcolors`, `\chessboardcolors` and `\arraycolor`.[27]
The names of these commands are inspired by the names of the commands provided by colortbl.

These commands don't color the cells which are in the "corners" if the key `corners` is used (cf. 5.3.3, p. 17).

[25] If you use Overleaf, Overleaf will do automatically a sufficient number of compilations.
[26] The available keys are `create-cell-nodes`, `sub-matrix` (and its subkeys) and `delimiters-color`.
[27] Remark that, in the `\CodeBefore`, PGF/TikZ nodes of the form "`(i-|j)`" are also available to indicate the position to the potential rules: cf. 15.3, p. 68.

These commands respect the rounded corners if the key `rounded-corners` (cf. 14.2, p. 59) has been used.

These commands also apply the colored panels between the rules of the double rules (that's a difference with colortbl).

All these commands accept an optional argument, between square brackets and in first position. That optional argument may contain two elements (separated by a comma)

- the colorimetric space (`RGB`, `rgb`, `HTML`, etc) as specified by the the extension xcolor;
- a specification of opacity of the form `opacity = value`.[28]

We describe now in detail those commands.

- The command `\cellcolor` takes its name from the command `\cellcolor` of colortbl.

  This command takes in as mandatory arguments a color and a **list of cells**, each of which with the format *i*-*j* where *i* is the number of the row and *j* the number of the column of the cell. In fact, despite its name, this command may be used to color a whole row (with the syntax *i*-) or a whole column (with the syntax -*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



- The command `\rectanglecolor` takes three mandatory arguments. The first is the color. The second is the upper-left cell of the rectangle and the third is the lower-right cell of the rectangle.

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

a	b	c
e	f	g
h	i	j



- The command `\arraycolor` takes in as mandatory argument a color and color the whole tabular with that color (except the potential exterior rows and columns: cf. 9, p. 36). It's only a particular case of `\rectanglecolor`.

- The command `\chessboardcolors` takes in as mandatory arguments two colors and it colors the cells of the tabular in quincunx with these colors.

[28] Caution: that feature creates instructions of transparency in the PDF and some readers of PDF don't support such instructions. The opacity is applied by using `\pgfsetfillopacity` of PGF.

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

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



We have used the key `r` which aligns all the columns rightwards (cf. 14.5, p. 60).

- The command `\rowcolor` takes its name from the command `\rowcolor` of colortbl. Its first mandatory argument is the color and the second is a comma-separated list of rows or interval of rows with the form *a*-*b* (an interval of the form *a*- represent all the rows from the row *a* until the end). Contrarily to the command `\rowcolor` of colortbl, that commands applies also the color between the rules of a double rule.

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

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



- The command `\columncolor` takes its name from the command `\columncolor` of colortbl. Its syntax is similar to the syntax of `\rowcolor`.

- The command `\rowcolors` (with a *s*) takes its name from the command `\rowcolors` of colortbl. The *s* emphasizes the fact that there is *two* colors. This command colors alternately the rows of the tabular with the two colors (provided in second and third argument), beginning with the row whose number is given in first (mandatory) argument. One of the arguments of color may be empty (no color is applied in the corresponding rows).

  In fact, the first (mandatory) argument is, more generally, a comma separated list of intervals describing the rows involved in the action of `\rowcolors` (an interval of the form *i*- describes in fact the interval of all the rows of the tabular, beginning with the row *i*).

  The last argument of `\rowcolors` is an optional list of pairs *key=value* (the optional argument in the first position corresponds to the colorimetric space). The available keys are `cols`, `restart` and `respect-blocks`.

  - The key `cols` describes a set of columns. The command `\rowcolors` will color only the cells of these columns. The value is a comma-separated list of intervals of the form *i*-*j* (where *i* or *j* may be replaced by `*`).
  - With the key `restart`, each interval of rows (specified by the first mandatory argument) begins with the same color.[29]

[29] Otherwise, the color of a given row relies only upon the parity of its absolute number.

– With the key `respect-blocks` the "rows" alternately colored may extend over several rows if they have to incorporate blocks (created with the command `\Block`: cf. 4, p. 6).

```
\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}
```

Résultats		
A	Pierre	12
	Jacques	8
B	Stéphanie	18
	Amélie	20
	Henri	14
	Estelle	15



- The extension nicematrix provides also a command `\rowlistcolors`. This command generalises the command `\rowcolors`: instead of two successive arguments for the colors, this command takes in an argument which is a (comma-separated) list of colors. In that list, the symbol `=` represents a color identical to the previous one.

```
\begin{NiceTabular}{c}[rounded-corners]
\CodeBefore
  \rowlistcolors{1}{red!15,blue!15,green!15}
\Body
  Peter \\
  James \\
  Abigail \\
  Elisabeth \\
  Claudius \\
  Jane \\
  Alexandra \\
\end{NiceTabular}
```

Peter
James
Abigail
Elisabeth
Claudius
Jane
Alexandra



It's also possible to use in the command `\rowlistcolors` a color series defined by the command `\definecolorseries` of xcolor (and initialized with the command `\resetcolorseries`[30]).

```
\begin{NiceTabular}{c}
\CodeBefore
  \definecolorseries{BlueWhite}{rgb}{last}{blue}{white}
  \resetcolorseries[\value{iRow}]{BlueWhite}
  \rowlistcolors{1}{BlueWhite!!+}
\Body
  Peter \\
  James \\
  Abigail \\
  Elisabeth \\
  Claudius \\
  Jane \\
  Alexandra \\
\end{NiceTabular}
```

Peter
James
Abigail
Elisabeth
Claudius
Jane
Alexandra



[30] For the initialization, in the following example, you have used the LaTeX counter `iRow` (which corresponds to the internal TeX counter `\c@iRow`) which, when used in the `\CodeBefore` (and in the `\CodeAfter`) corresponds to the number of rows of the array: cf. 14.8, p. 62. That leads to an adjustment of the gradation of the colors to the size of the tabular.

We recall that all the color commands that we have described don't color the cells which are in the "corners". In the following example, we use the key `corners` to require the determination of the "corner" *east* (E).

```
\begin{NiceTabular}{ccccccc}[corners=E,margin,hvlines,first-row,first-col]
\CodeBefore
  \rowlistcolors{1}{blue!15, }
\Body
    & 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 \\
\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



The previous example uses the keys `first-row` and `first-col` which are described in the chapter concerning the "exterior" rows and columns (cf. 9, p. 36).
As can be seen, *by default*, the coloring commands that we have described don't apply in those exterior rows and columns.
However, it may still be possible to color in those rows and columns by providing explicitly the numbers of those rows and columns.
In the following example, we require a color in the column 0 (which is the "first column" and which exists because the key `first-col` has been used).

```
\begin{NiceTabular}{ccccccc}[corners=E,margin,hvlines,first-row,first-col]
\CodeBefore
  \rowlistcolors{1}{blue!15, }
  \columncolor{red!15}{0}
\Body
    & 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 \\
\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



One should remark that all the previous commands are compatible with the commands of booktabs (`\toprule`, `\midrule`, `\bottomrule`, etc). However, booktabs is *not* loaded by nicematrix.

```
\begin{NiceTabular}[c]{lSSSS}[no-cell-nodes] % no-cell-nodes is optional
\CodeBefore
  \rowcolor{red!15}{1-2}
  \rowcolors{3}{blue!15}{}
\Body
  \toprule
  \Block[C]{2-1}{Product} &
  \Block{1-3}{dimensions (cm)} & & &
  \Block{2-1}{\rotate Price} \\
  \cmidrule(rl){2-4}
   & L & l & h \\
  \midrule
  small    & 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  \\
  special  & 12  & 12   & 0.5 & 70    \\
  \bottomrule
\end{NiceTabular}
```

Product	dimensions (cm)			Price
	L	l	h	
small	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
special	12	12	0.5	70



We have used the type of column `S` of siunitx (which should be loaded by the user).

It's also possible, in the `\CodeBefore`, to use the commands `\EmptyColumn` and `\EmptyRow`. The command `\EmptyColumn` takes in as argument a (comma-separated) list of numbers of columns and requires that no rule nor background colors will be drawn in the corresponding columns. The command `\EmptyRow` is similar for the rows.

```
\begin{NiceTabular}{ccccc}[hvlines,no-cell-nodes] % no-cell-nodes is optional
\CodeBefore
  \rowcolor{blue!15}{1}
  \EmptyColumn{3}
\Body
  one & two && three & four \\
  one & \Block{}{two\\ rows} && three & four \\
\end{NiceTabular}
```

one	two	three	four
one	two rows	three	four



These commands may typically be used in order to compose several tabulars side by side. If you want to draw double-rules, you should instead consider the command `\FalseRow` and the letter `F` (cf. 5.3.8, p. 24).

6.3 Color tools to be used inside the tabular

The extension nicematrix also provides commands to be used directly in the tabular (as does colortbl). These available commands are the following ones (the first three ones are inspired by the similar commands of colortbl).

- `\cellcolor` which colorizes a cell;[31]
- `\rowcolor` which must be used in a cell and which colorizes the end of the row;[32]
- `\columncolor` which must be used in the preamble of the environment with the same syntax as the corresponding command of colortbl (however, unlike the command `\columncolor` of

[31] That command `\cellcolor` will delete the following spaces (which does not the command `\cellcolor` of colortbl). Moreover, if one wishes to define a command above that command `\cellcolor`, it must be protected in the TeX sens (whereas, if it were the command `\cellcolor` of colortbl, one should, on the contrary, write a *fully expandable* command).

[32] If you want a command to color the following n rows, consider the command `\RowStyle` and its key `fillcolor`: cf. 7, p. 32

colortbl, this command `\columncolor` can appear within another command, itself used in the preamble of the array; on the other hand, the two final optional arguments for fixing offset are not supported);

- `\rowcolors` which takes in as arguments two colors and color the rest of the tabular with those colors (must be used after a potential `\hline`, `\Hline` or `\toprule`!) ;
- `\rowlistcolors` which takes in as argument a color and color the rest of the tabular with the colors of that list of colors.[33]

These commands are compatible with the commands for the overlays of Beamer (`\only`, etc.)

```
\NewDocumentCommand { \Blue } { } { \columncolor{blue!15} }
\begin{NiceTabular}{>{\Blue}c>{\Blue}cc}
\toprule
  \rowcolor{red!15}
  Last name & First name & Birth day \\
\midrule
  Achard  & Jacques & 5 juin 1962 \\
  Lefebvre & Mathilde & 23 mai 1988 \\
  Vanesse & Stephany & 30 octobre 1994 \\
  Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}
```

Last name	First name	Birth day
Achard	Jacques	5 juin 1962
Lefebvre	Mathilde	23 mai 1988
Vanesse	Stephany	30 octobre 1994
Dupont	Chantal	15 janvier 1998

Each use of the `\rowlistcolors` (or `\rowcolors`, which is, in fact, a special case of `\rowlistcolors`) stops the potential coloring schemes[34] specified by a previous command `\rowlistcolors`. In particular, it's possible to start coloring the rows with `\rowlistcolors{...}` and stop coloring by a command `\rowlistcolors` with an empty argument.

```
\begin{NiceTabular}{c}[hvlines]
  one \\
  two \\
\rowlistcolors{red!15}
  three \\
  four \\
  five \\
\rowlistcolors{}
  six \\
  seven \\
\end{NiceTabular}
```

one
two
three
four
five
six
seven

Here is an example of use of `\rowcolors` in conjunction with `create-blocks-in-col`.

[33] When the command `\rowlistcolors` (or the command `\rowcolors`) is used in a cell of the column j of the array, the command applies only on the columns above j (by design).

[34] We say *schemes* in plural form because it's possible to start simultaneously several coloring schemes if they apply on different columns.

```
\begin{NiceTabular}{lr}[hvlines,create-blocks-in-col=1]
\rowcolors{blue!10}{}[respect-blocks]
  John      & 12 \\
            & 13 \\
  Steph     &  8 \\
  Sarach    & 18 \\
            & 17 \\
            & 15 \\
  Ashley    & 20 \\
  Henry     & 14 \\
  Madison   & 15 \\
            & 19
\end{NiceTabular}
```

John	12
	13
Steph	8
Sarach	18
	17
	15
Ashley	20
Henry	14
Madison	15
	19



6.4 The special color "nocolor"

The extension nicematrix provides the special color `nocolor` which may be used in all the coloring commands provided by nicematrix (in the `\CodeBefore` or in the array itself).
The cells marked by this color won't be colored, whatever the other instructions of coloring which may apply to these cells.
Therefore, the color `nocolor` is an easy way to add an exception to a more general coloring command.

7 The command `\RowStyle`

The command `\RowStyle` takes in as argument some formatting instructions that will be applied to each cell on the rest of the current row.

That command also takes in as optional argument (between square brackets) a list of *key=value* pairs.

- The key `nb-rows` sets the number of rows to which the specifications of the current command will apply (with the special value `*`, it will apply to all the following rows).
- The keys `cell-space-top-limit`$^+$, `cell-space-bottom-limit`$^+$ and `cell-space-limits`$^+$ are available with the same meaning that the corresponding global keys (cf. 2, p. 4).
- The key `fill` (alias: `rowcolor`) sets the color of the background and `opacity`[35] sets its opacity.

  These commands have a normal syntax but also an "additive" syntax. That means that it's possible to write, for example, `cell-space-limits = 2pt` but also `cell-space-limits += 1pt`.
- If the key `rounded-corners` is used, that background will have rounded corners.
- The key `color` sets the color of the text.[36]
- The key `bold` enforces bold characters for the cells of the row, both in math and text mode.

[35] Caution: that feature creates instructions of transparency in the PDF and some readers of PDF don't support such instructions. The opacity is applied by using `\pgfsetfillopacity`.

[36] The key `color` uses the command `\color` but also inserts an instruction `\leavevmode` before. This instruction prevents an extra vertical space in the cells which belong to columns of type `p`, `b`, `m` and `X` (which start in vertical mode of LaTeX). For the columns `V` (of varwidth), that's not enough, except when LuaLaTeX is used with the package luacolor (see question 460489 on TeX StackExchange).

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

first	second	third	fourth
1	2	3	4
I	II	III	IV

The command `\rotate` is described in the section 14.6, p. 61.

8 The width of the columns

8.1 Basic tools

In the environments with an explicit preamble (like `{NiceTabular}`, `{NiceArray}`, etc.), it's possible to fix the width of a given column with the standard letters `w`, `W`, `p`, `b` and `m` of the package array (which is loaded by 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

In the environments of nicematrix, it's also possible to fix the *minimal* width of all the columns (except the potential exterior columns: cf. 9, p. 36) directly with the key `columns-width`.

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

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

Note that the space inserted between two columns (equal to 2 `\tabcolsep` in `{NiceTabular}` and to 2 `\arraycolsep` in the other environments) is not suppressed (of course, it's possible to suppress this space by setting `\tabcolsep` or `\arraycolsep` equal to 0 pt before the environment).

It's possible to give the special value `auto` to the option `columns-width`: all the columns of the array will have a width equal to the widest cell of the array.[37]

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

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

Without surprise, it's possible to fix the minimal width of the columns of all the arrays of a current scope with the command `\NiceMatrixOptions`.

[37] The result is achieved with only one compilation (but PGF/TikZ will have written information in the aux file and a message requiring a second compilation will appear).

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

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



It's also possible to fix a zone where all the matrices will have their columns of the same width, equal to the widest cell of all the matrices. This construction uses the environment `{NiceMatrixBlock}` with the option `auto-columns-width`[38]. The environment `{NiceMatrixBlock}` has no direct link with the command `\Block` presented previously in this document (cf. 4, p. 6).

```
\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}
```

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

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



New 7.13

One may wish to create a tabular with actual square cells. In this aim, nicematrix provides for the key `columns-width` a special value called `row-height`. With that value, the columns will have a minimal absolute[39] width equal to the total height (`\dp` + `\ht`) of the strut used by LaTeX to set the height of the rows. With the extension array (which is loaded by nicematrix), that strut may be customized with the `\arraystretch` and `\extrarowheight`.

```
\renewcommand{\arraystretch}{1.5}
$\begin{NiceArray}{cccc}[hvlines,columns-width=row-height]
a & b & c & d \\
b & c & d & a \\
c & d & a & b \\
d & a & b & c
\end{NiceArray}$
```

a	b	c	d
b	c	d	a
c	d	a	b
d	a	b	c



8.2 The columns V of varwidth

Let's recall first the behaviour of the environment `{varwidth}` of the eponymous package varwidth. That environment is similar to the classical environment `{minipage}` but the width provided in the argument is only the *maximal* width of the created box. In the general case, the width of the box constructed by an environment `{varwidth}` is the natural width of its contents.

That point is illustrated on the following examples.

[38] At this time, this is the only usage of the environment `{NiceMatrixBlock}` but it may have other usages in the future.

[39] We have use *absolute* to mean that the inter-columns space (set by `\arraycolsep` or `\tabcolsep`) is set to 0 pt by this option.

```
\fbox{%
\begin{varwidth}{8cm}
\begin{itemize}
\item first item
\item second item
\end{itemize}
\end{varwidth}}
```

- first item
- second item



```
\fbox{%
\begin{minipage}{8cm}
\begin{itemize}
\item first item
\item second item
\end{itemize}
\end{minipage}}
```

- first item
- second item



The package varwidth provides also the column type `V`. A column of type `V{`⟨*dim*⟩`}` encapsulates all its cells in a `{varwidth}` with the argument ⟨*dim*⟩ (and does also some tuning).

When the package varwidth is loaded, the columns `V` of varwidth are supported by nicematrix.

```
\begin{NiceTabular}[corners=NW,hvlines]{V{3cm}V{3cm}V{3cm}}
  & some text & some very very very long text \\
  some very very very long text \\
  some very very very long text
\end{NiceTabular}
```



	some text	some very very very long text
some very very very long text		
some very very very long text		

Concerning nicematrix, one of the benefits of this type of columns is that, for a cell of a column of type `V`, the PGF/TikZ node created by nicematrix for the content of that cell has a width adjusted to the content of the cell : cf. 15.1.2, p. 66.

The columns `V` of nicematrix supports the keys `t`, `p`, `m`, `b`, `l`, `c` and `r` also supported by the columns `X`: see their description in the section 8.3, p. 35.

One should remark that the extension varwidth (at least in its version 0.92) has some problems: for instance, with LuaLaTeX, it does not work when the content begins with `\color`. Moreover, varwidth must be loaded after the package array (itself loaded by nicematrix).

8.3 The columns X

The environment `{NiceTabular}` provides `X` columns similar to those provided by the environment `{tabularx}` of the eponymous package.[40]

The required width of the tabular may be specified with the key `width` (in `{NiceTabular}` or in `\NiceMatrixOptions`). The initial value of this parameter is the parameter `\linewidth` of LaTeX (and not `\textwidth`).

[40] The environment `{tabularx}` catches its body as an argument and typesets it several times during a LaTeX run in order to compute the width of the columns. On its side, `{NiceTabular}` does not catch its body as an argument (except when the key `light-syntax` is in force) and typesets the tabular only once but writes informations on the `aux` file, and requires several compilations.

For sake of similarity with the environment `{tabularx}`, nicematrix also provides an environment `{NiceTabularX}` with a syntax similar to the syntax of `{tabularx}`, that is to say with a first mandatory argument which is the width of the tabular.

The specifier `X` takes in an optional argument (between square brackets) which is a list of keys.

- It's possible to give a weight for the column by providing a positive number directly as argument of the specifier `X`. For example, a column `X[0.5]` will have a width equal to the half of the width of a column `X` (which has a weight equal to 1).
- It's possible to specify an horizontal alignment with one of the letters `l`, `c` and `r` (which insert respectively `\raggedright`, `\centering` and `\raggedleft` followed by `\arraybackslash`).[41]
- It's possible to specify a vertical alignment with one of the keys `t` (alias `p`), `m` and `b` (which construct respectively columns of type `p`, `m` and `b`). The initial value is `t`.
- It's possible to use the key `V` in a column of type `X`. When that key is used, the column `X` behaves in fact like a column `V` of the extension varwidth (which must have been loaded by the user). Henece, the width of the columns, as computed by the `X` process, is in fact the *maximal* width of the column.

```
\begin{NiceTabular}[width=9cm]{X[c,m]X[0.5,c,m]}[hvlines]
  a rather long text which fits on several lines
  & a rather long text which fits on several lines \\
  a shorter text & a shorter text
\end{NiceTabular}
```

a rather long text which fits on several lines	a rather long text which fits on several lines
a shorter text	a shorter text

9 The exterior rows and columns

The keys `first-row`, `last-row`, `first-col` and `last-col` allow the composition of exterior rows and columns in the environments of nicematrix. It's particularly interesting for the (mathematical) matrices.
A potential "first row" (exterior) has the number 0 (and not 1). Idem for the potential "first column".

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

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

[41] In fact, when ragged2e is loaded, nicematrix uses the commands `\RaggedRight`, `\Centering` and `\RaggedLeft` of ragged2e instead of the commands `\raggedright`, `\centering` and `\raggedleft`. That ensures a better output.

The dotted lines of this example have been drawn with the tools presented in the section 10, p. 38.

We have several remarks to do.

- For the environments with an explicit preamble (i.e. `{NiceTabular}`, `{NiceArray}` and its variants), no letter must be given in that preamble for the potential first column and the potential last column: they will automatically (and necessarily) be of type `r` for the first column and `l` for the last one.[42]
- One may wonder how nicematrix determines the number of rows and columns which are needed for the composition of the "last row" and "last column".
  - For the environments with explicit preamble, like `{NiceTabular}` and `{pNiceArray}`, the number of columns can obviously be computed from the preamble.
  - When the option `light-syntax` (cf. 14.9, p. 63) is used, nicematrix has, in any case, to load the whole body of the environment (and that's why it's not possible to put verbatim material in the array with the option `light-syntax`). The analysis of this whole body gives the number of rows and the number of columns.
  - In the other cases, nicematrix compute the number of rows and columns during the first compilation and write the result in the aux file for the next run.

    However, it's possible to provide the number of the last row and the number of the last column as values of the options `last-row` *and* `last-col`*, tending to an acceleration of the whole compilation of the document.* That's what we will do throughout the rest of the document.

It's possible to control the appearance of these rows and columns with options `code-for-first-row`$^+$, `code-for-last-row`$^+$, `code-for-first-col`$^+$ and `code-for-last-col`$^+$. These options specify tokens that will be inserted before each cell of the corresponding row or column.
Those keys have a normal syntax but also an "additive" syntax. For example, it's possible to write `code-for-first-row = \color{blue}` but also `code-for-first-row += \color{blue}` (in the latter case, the tokens are appended on the right of the current value of the parameter).

```
\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}}

$\begin{pNiceArray}{cc|cc}[first-row,last-row=5,first-col,last-col,xdots/nullify]
       & C_1    & \multicolumn1c{\Cdots} &        & C_4    &        \\
L_1    & a_{11} & a_{12} & a_{13} & a_{14} & L_1    \\
\Vdots & a_{21} & a_{22} & a_{23} & a_{24} & \Vdots \\
\hline
       & a_{31} & a_{32} & a_{33} & a_{34} &        \\
L_4    & a_{41} & a_{42} & a_{43} & a_{44} & L_4    \\
       & C_1    & \multicolumn1c{\Cdots} &        & C_4    &
\end{pNiceArray}$
```

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

[42] The users wishing exterior columns with another type of alignment should consider the command `\SubMatrix` available in the `\CodeAfter` and in the `\CodeBefore` (cf. 12.2, p. 48) or try to put delimiter directly in the preamble of the array (cf. 11, p. 45).

Remarks

- As shown in the previous example, the horizontal and vertical rules don't extend in the exterior rows and columns. This remark also applies to the customized rules created by the key `custom-line` (cf. 5.3.5, p. 19).
- A specification of color present in `code-for-first-row` also applies to a dotted line drawn in that exterior "first row" (except if a value has been given to `xdots/color`). Idem for the other exterior rows and columns.
- Logically, the potential option `columns-width` (cf. 8, p. 33) doesn't apply to the "first column" and "last column".
- For technical reasons, it's not possible to use the option of the command `\\` after the "first row" or before the "last row". The placement of the delimiters would be wrong. If you are looking for a workaround, consider the command `\SubMatrix` in the `\CodeAfter` (and the `\CodeBefore`) described in the section 12.2, p. 48.

10 Dotted leaders in the matrices

Inside the environments of the package nicematrix, new commands are defined: `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots` and `\Iddots`. These commands are intended to be used in place of `\dots`, `\cdots`, `\vdots`, `\ddots` and `\iddots`.[43]

Each of them must be used alone in the cell of the array and it draws a dotted line as leader between the first non-empty cells[44] on both sides of the current cell. Of course, for `\Ldots` and `\Cdots`, it's an horizontal line; for `\Vdots`, it's a vertical line and for `\Ddots` and `\Iddots` diagonal ones. It's possible to change the color of these lines with the option `color`.[45]

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

$$\begin{bmatrix} a_1 & \cdots & \cdots & a_1 \\ \vdots & & & \\ & a_2 & \cdots & a_2 \\ \vdots & & \ddots & \\ a_1 & a_2 & & a_n \end{bmatrix}$$

In order to represent the null matrix, one can use the following code:

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

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

However, one may prefer a larger matrix. Usually, in such a case, the users of LaTeX add a new row and a new column. It's possible to use the same method with nicematrix:

[43] The command `\iddots`, defined in nicematrix, is a variant of `\ddots` with dots going forward. If `mathdots` is loaded, the version of `mathdots` is used. It corresponds to the command `\adots` of unicode-math.

[44] The precise definition of a "non-empty cell" is given below (cf. 17.2, p. 72).

[45] It's also possible to change the color of all these dotted lines with the option `xdots/color` (*xdots* to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.): cf. 10.5, p. 42.

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

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

In the first column of this example, there are two instructions `\Vdots` but, of course, only one dotted line is drawn.

In fact, in this example, it would be possible to draw the same matrix more easily with the following code:

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

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

There are also other means to change the size of the matrix. Someone might want to use the optional argument of the command `\\` for the vertical dimension and a command `\hspace*` in a cell for the horizontal dimension.[46]
However, a command `\hspace*` might interfere with the construction of the dotted lines. That's why the package nicematrix provides a command `\Hspace` which is a variant of `\hspace` transparent for the dotted lines of nicematrix.

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

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

10.1 The option xdots/nullify

Consider the following matrix composed classicaly with the environment `{pmatrix}` of amsmath.

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

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

If we add `\ldots` instructions in the second row, the geometry of the matrix is modified.

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

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

By default, with nicematrix, if we replace `{pmatrix}` by `{pNiceMatrix}` and `\ldots` by `\Ldots`, the geometry of the matrix is not changed.

[46] In nicematrix, one should use `\hspace*` and not `\hspace` for such an usage because nicematrix loads array. One may also remark that it's possible to fix the width of a column by using the environment `{NiceArray}` (or one of its variants) with a column of type `w` or `W`: cf. 8, p. 33

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

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



However, one may prefer the geometry of the first matrix A and would like to have such a geometry with a dotted line in the second row. It's possible by using the option `xdots/nullify`[47] (and only one instruction `\Ldots` is necessary).

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

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



The option `xdots/nullify` smashes the instructions `\Ldots` (and the variants) horizontally but also vertically.

Caution: the key `xdots/nullify` has a name that may be confusing; that key does not imply that the dotted rules won't be drawn!

10.2 The commands \Hdotsfor and \Vdotsfor

Some people commonly use the command `\hdotsfor` of amsmath in order to draw horizontal dotted lines in a matrix. In the environments of nicematrix, one should use instead `\Hdotsfor` in order to draw dotted lines similar to the other dotted lines drawn by `\Cdots`, `\Vdots`, etc.
As with the other commands of nicematrix (like `\Cdots`, `\Ldots`, `\Vdots`, etc.), the dotted line drawn with `\Hdotsfor` extends until the contents of the cells on both sides.

```
$\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 \\
1 & \Hdotsfor{3} & 5 \\
1 & 2 & 3 & 4 & 5 \\
1 & 2 & 3 & 4 & 5
\end{pNiceMatrix}$
```

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



However, if these cells are empty, the dotted line extends only in the cells specified by the argument of `\Hdotsfor` (by design).

```
$\begin{pNiceMatrix}
1 & 2 & 3 & 4 & 5 \\
  & \Hdotsfor{3} \\
1 & 2 & 3 & 4 & 5 \\
1 & 2 & 3 & 4 & 5
\end{pNiceMatrix}$
```

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



Remark: Unlike the command `\hdotsfor` of amsmath, the command `\Hdotsfor` may be used even when the package colortbl[48] is loaded (however, with nicematrix, loading colortbl is rather point-less since nicematrix provides its own coloring tools: cf. 6.2, p. 25).

The package nicematrix also provides a command `\Vdotsfor` similar to `\Hdotsfor` but for the vertical dotted lines.

The following example uses both `\Hdotsfor` and `\Vdotsfor`:

[47] The key `xdots/nullify` has an alias, which is `nullify-dots`.
[48] We recall that when xcolor is loaded with the option `table`, the package colortbl is loaded.

```
$\begin{bNiceMatrix}
C[a_1,a_1] & \Cdots & C[a_1,a_n]
  & \hspace*{20mm} & C[a_1,a_1^{(p)}] & \Cdots & C[a_1,a_n^{(p)}] \\
\Vdots & \Ddots & \Vdots
  & \Hdotsfor{1} & \Vdots & \Ddots & \Vdots \\
C[a_n,a_1] & \Cdots & C[a_n,a_n]
  & & C[a_n,a_1^{(p)}] & \Cdots & C[a_n,a_n^{(p)}] \\
\rule{0pt}{15mm}\NotEmpty  & \Vdotsfor{1} & & \Ddots & & \Vdotsfor{1} \\
C[a_1^{(p)},a_1] & \Cdots & C[a_1^{(p)},a_n]
  & & C[a_1^{(p)},a_1^{(p)}] & \Cdots & C[a_1^{(p)},a_n^{(p)}] \\
\Vdots & \Ddots & \Vdots
  & \Hdotsfor{1} & \Vdots & \Ddots & \Vdots \\
C[a_n^{(p)},a_1] & \Cdots & C[a_n^{(p)},a_n]
  & & C[a_n^{(p)},a_1^{(p)}] & \Cdots & C[a_n^{(p)},a_n^{(p)}]
\end{bNiceMatrix}$
```

$$\begin{bmatrix}
C[a_1,a_1] & \cdots & C[a_1,a_n] & & C[a_1,a_1^{(p)}] & \cdots & C[a_1,a_n^{(p)}] \\
\vdots & \iddots & \vdots & \cdots\cdots & \vdots & \iddots & \vdots \\
C[a_n,a_1] & \cdots & C[a_n,a_n] & & C[a_n,a_1^{(p)}] & \cdots & C[a_n,a_n^{(p)}] \\
 & & & \Ddots & & & \\
C[a_1^{(p)},a_1] & \cdots & C[a_1^{(p)},a_n] & & C[a_1^{(p)},a_1^{(p)}] & \cdots & C[a_1^{(p)},a_n^{(p)}] \\
\vdots & \iddots & \vdots & \cdots\cdots & \vdots & \iddots & \vdots \\
C[a_n^{(p)},a_1] & \cdots & C[a_n^{(p)},a_n] & & C[a_n^{(p)},a_1^{(p)}] & \cdots & C[a_n^{(p)},a_n^{(p)}]
\end{bmatrix}$$

10.3 How to generate the dotted leaders transparently

Imagine you have a document with a great number of mathematical matrices with ellipsis. You may wish to use the dotted lines of nicematrix without having to modify the code of each matrix. It's possible with the keys. `renew-dots` and `renew-matrix`.[49]

- The option `renew-dots`

  With this option, the commands `\ldots`, `\cdots`, `\vdots`, `\ddots`, `\iddots`[43] and `\hdotsfor` are redefined within the environments provided by nicematrix and behave like `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots` and `\Hdotsfor`; the command `\dots` ("automatic dots" of `amsmath`) is also redefined to behave like `\Ldots`.

- The option `renew-matrix`

  With this option, the environment `{matrix}` is redefined and behave like `{NiceMatrix}`, and so on for the five variants with delimiters.

Therefore, with the keys `renew-dots` and `renew-matrix`, a classical code gives directly the output of nicematrix.

```
\NiceMatrixOptions{renew-dots,renew-matrix}
$\begin{pmatrix}
1      & \cdots & \cdots & 1      \\
0      & \ddots &        & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
0      & \cdots & 0      & 1
\end{pmatrix}$
```

$$\begin{pmatrix}
1 & \cdots & \cdots & 1 \\
0 & \ddots & & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & 1
\end{pmatrix}$$

[49] The options `renew-dots`, `renew-matrix` can be fixed with the command `\NiceMatrixOptions` like the other options. However, they can also be fixed as options of the command `\usepackage`.

However, one should note that the environments of nicematrix can't be nested, and, therefore, it's not possible to use `renew-matrix` if we have, for example, an environment `{pmatrix}` nested within another environment `{pmatrix}`.

10.4 The labels of the dotted leaders

The commands `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and `\Vdotsfor` (and the command `\line` in the `\CodeAfter` which is described in the section 12.1, p. 47) accept three optional arguments specified by the tokens `_`, `^` and `:` for labels positioned below, above or on the line. The arguments are composed in math mode with `\scriptstyle`.
The label specified by "`:`" is composed on a white background which is put on the line previously drawn (cf. example 18.9, p. 80).

```
$\begin{bNiceMatrix}
1 & \hspace*{1cm}             & 0 \\[8mm]
  & \Ddots^{n \text{ times}} &   \\
0 &                           & 1
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 1 & & 0 \\ & \overset{n \text{ times}}{\iddots} & \\ 0 & & 1 \end{bmatrix}$$

With the key `horizontal-label`, the label stays horizontal.

```
$\begin{bNiceMatrix}
1 & \hspace*{1cm}             & 0 \\[8mm]
  & \Ddots[horizontal-label]^{n \text{ times}} &   \\
0 &                           & 1
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 1 & & 0 \\ & \overset{n \text{ times}}{\iddots} & \\ 0 & & 1 \end{bmatrix}$$

10.5 Customisation of the dotted leaders

The dotted lines drawn by `\Ldots`, `\Cdots`, `\Vdots`, `\Ddots`, `\Iddots`, `\Hdotsfor` and `\Vdotsfor` (and by the command `\line` in the `\CodeAfter` which is described in the section 12.1, p. 47) may be customized by the following options (specified between square brackets after the command):

- `nullify`;
- `horizontal-label(s)`;
- `color`;
- `radius`;
- `shorten-start`$^+$, `shorten-end`$^+$ and `shorten`$^+$;
- `inter`;
- `number-of-dots`;
- `line-style`.

For an individual command, it's possible to write `horizontal-label` (in the singular).

These options may also be fixed with `\NiceMatrixOptions`, as options of `\CodeAfter` or at the level of a given environment but, in those cases, they must be prefixed by `xdots` (*xdots* to remind that it works for `\Cdots`, `\Ldots`, `\Vdots`, etc.), and, thus have for names:

- `xdots/nullify`;
- `xdots/horizontal-labels`;
- `xdots/color`;
- `xdots/radius`;

- `xdots/shorten-start`$^+$, `xdots/shorten-end`$^+$ and `xdots/shorten`$^+$;
- `xdots/inter`;
- `xdots/number-of-dots`;
- `xdots/line-style`.

For the clarity of the explanations, we will use those names.

The key `xdots/nullify` has yet been described in the part 10.1, p. 39.

With the key `xdots/horizontal-labels`, the labels (introduced by `_`, `^` and `:`) stay horizontal.

The option `xdots/color` fixes the color of the dotted line. It's possible to use a color defined "on the fly" (e.g.: `xdots/color = { [RGB]{204,204,255} }`). One should remark that the dotted lines drawn in the exterior rows and columns have a special treatment: cf. 9, p. 36.

The option `xdots/radius` fixes the radius of the dots. The initial value is 0.53 pt.

The keys `xdots/shorten-start`$^+$ and `xdots/shorten-end`$^+$ fix the margin at the extremities of the line. The key `xdots/shorten`$^+$ fixes both parameters. The initial value is 0.3 em.[50]

These keys have a normal syntax but also a "additive" syntax. That means that it's possible to write, for example, `xdots/shorten = 0.6em` but also `xdots/shorten += 3pt` (ant we remind that the initial value is *not* zero!).

The option `xdots/inter` fixes the length between the dots. The initial value is 0.45 em (it is recommended to use a unit of length dependent of the current font).

New 7.13

The key `xdots/number-of-dots` fixes the number of dots, independently of the distance between the cells to rely.

The option `xdots/line-style`

It should be pointed that, by default, the lines drawn by TikZ with the parameter `dotted` are composed of square dots (and not rounded ones).[51]

```
\tikz \draw [dotted] (0,0) -- (5,0) ;
```

In order to provide lines with rounded dots in the style of those provided by `\ldots` (at least with the *Computer Modern* fonts), the package nicematrix embeds its own system to draw a dotted line (and this system uses PGF and not TikZ). This style is called `standard` and that's the initial value of the parameter `xdots/line-style`.
However (when TikZ is loaded) it's possible to use for `xdots/line-style` any style provided by TikZ, that is to say any sequence of options provided by TikZ for the TikZ paths (with the exception of "`color`", "`shorten >`" and "`shorten <`").

Here is for example a tridiagonal matrix with the style `loosely dotted`:

[50] In fact, when `xdots/shorten`, `xdots/shorten-start` and `xdots/shorten-end` are used in `\NiceMatrixOptions` or at the level of an environment (such as `{pNiceMatrix}`), those keys only apply to the extremeties of dotted lines corresponding to a non-empty content of a cell. When they are used for a command such as `\Cdots` (and, in that case, their names are `shorten`, `shorten-start` and `shorten-end`), they apply to all the extremities.

[51] The first reason of this behaviour is that the PDF format includes a description for dashed lines. The lines specified with this descriptor are displayed very efficiently by the PDF readers. It's easy, starting from these dashed lines, to create a line composed by square dots whereas a line of rounded dots needs a specification of each dot in the PDF file. Nevertheless, you can have a look at the following page to see how to have dotted rules with rounded dots in TikZ: `https://tex.stackexchange.com/questions/52848/tikz-line-with-large-dots`

```
$\begin{pNiceMatrix}[xdots={nullify, shorten += 2pt, line-style = loosely dotted}]
a      & b      & 0      &        & \Cdots & 0      \\
b      & a      & b      & \Ddots &        & \Vdots \\
0      & b      & a      & \Ddots &        &        \\
       & \Ddots & \Ddots & \Ddots &        & 0      \\
\Vdots &        &        &        &        & b      \\
0      & \Cdots &        & 0      & b      & a
\end{pNiceMatrix}$
```

$$\begin{pmatrix} a & b & 0 & \cdots & \cdots & 0 \\ b & a & b & \ddots & & \vdots \\ 0 & b & a & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & & 0 \\ \vdots & & \ddots & & & b \\ 0 & \cdots & \cdots & 0 & b & a \end{pmatrix}$$

10.6 The dotted leaders and the rules

The dotted lines determine virtual blocks which have the same behaviour regarding the rules (the rules specified by the specifier `|` in the preamble, by the command `\Hline`, by the keys `hlines`, `vlines`, `hvlines` and `hvlines-except-borders` and by the tools created by `custom-line` are not drawn within the blocks).[52]

```
$\begin{bNiceMatrix}[margin,hvlines]
\Block{3-3}<\LARGE>{A} & & & 0 \\
& \hspace*{1cm} & & \Vdots \\
& & & 0 \\
0 & \Cdots& 0 & 0
\end{bNiceMatrix}$
```

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

10.7 The commands `\Hbrace` and `\Vbrace`

Since, as said previously, it's possible to use, with the commands `\Cdots`, `\Ldots`, `\Vdots`, etc. all the styles of lines provided by TikZ, one may wish to use those commands to draw braces with the decoration `brace` provided by the library decorations.pathreplacing of TikZ.

In order to facilitate that use, nicematrix provides two commands `\Hbrace` and `\Vbrace`. Those commands are available only if TikZ has been loaded with its library decorations.pathreplacing (before or after the loading of nicematrix). Otherwise, a (non-fatal) error will be raised.

```
\usepackage{tikz}
\usetikzlibrary{decorations.pathreplacing}
```

The commands `\Hbrace` and `\Vbrace` have the same syntax. Both commands take in three arguments:

- an optional argument, between square brackets, which contains a list of *key=value* pairs: the keys allowed are `color`, `horizontal-labels`, `shorten`, `shorten-start` and `shorten-end`.

  A key `brace-shift`$^+$ is also provided to shift the brace (and its label) towards the exterior of the matrix. That key has a normal syntax but also an "additive" syntax, which means that it's possible to write, for example, `brace-shift = 3pt` but also `brace-shift += 1pt`.

- a mandatory argument which is the number of columns (for `\Hbrace`) or the number of rows (for `\Vbrace`) over which the command applies;

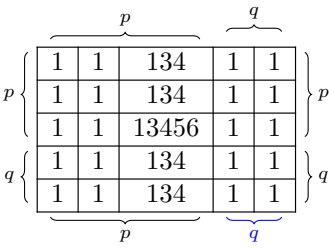
- an mandatory argument which is the label of the curly brace.

[52] On the other side, the command `\line` in the `\CodeAfter` (cf. 12.1, p. 47) does *not* create any block.

As regards the ampersands (&), \Hbrace has the same behavior as the commands \multicolumn, \hdotsfor, \Hdotsfor (provided by nicematrix), etc.: only one ampersand must be inserted, even if the brace that will be drawn will extend on several columns.

```
\usepackage{tikz}                          % in the preamble
\usetikzlibrary{decorations.pathreplacing} % in the preamble

$\begin{NiceArray}{ccccc}%
  [ hvlines ,
    first-row ,
    last-row = 6,
    first-col ,
    last-col ,
    xdots/horizontal-labels ]
& \Hbrace{3}{p} & \Hbrace[brace-shift=1mm]{2}{q} \\
\Vbrace{3}{p} & 1 & 1 & 134 & 1 & 1 & \Vbrace{3}{p} \\
& 1 & 1 & 134 & 1 & 1 \\
& 1 & 1 & 13456 & 1 & 1 \\
\Vbrace{2}{q} & 1 & 1 & 134 & 1 & 1 & \Vbrace{2}{q}\\
& 1 & 1 & 134 & 1 & 1 \\
& \Hbrace{3}{p} & \Hbrace[color=blue]{2}{q} \\
\end{NiceArray}$
```



The TikZ style used by nicematrix to draw those braces is called `nicematrix/brace`. Here is the initial value of that style:

```
decoration = { brace , raise = -0.15 em } ,
decorate
```

The final user can change that TikZ style as he wishes (respecting the structure with the keys `decorate` and `decoration`).

For another example of use of \Hbrace and \Vbrace, see 18.8 p. 79.

It's also possible to draw braces of the mathematical mode of LaTeX by using the commands \OverBrace and \UnderBrace (cf. 12.3, p. 50) and the command \SubMatrix (cf. 12.2, p. 48) in the \CodeAfter.

11 Delimiters in the preamble of the environment

In the environments with preamble ({NiceArray}, {pNiceArray}, etc.), it's possible to put vertical delimiters directly in the preamble of the environment.[53]

The opening delimiters should be prefixed by the keyword `\left` and the closing delimiters by the keyword `\right`. It's not mandatory to use \left and \right pair-wise.

All the vertical extensible delimiters of LaTeX are allowed.

Here is a example which uses the delimiters \lgroup and \rgroup.

```
$\begin{NiceArray}{\left\lgroup ccc\right\rgroup l}
1 & 2 & 3 &
4 & 1 & 6 &
7 & 8 & 9 & \scriptstyle L_3 \gets L_3 + L_1 + L_2
\end{NiceArray}$
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 1 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{matrix} \\ \\ L_3 \leftarrow L_3+L_1+L_2 \end{matrix}$$

[53] This syntax is inspired by the extension blkarray.

For this example, it would also have been possible to use the environment `{NiceArrayWithDelims}` (cf. 14.11, p. 64) and the key `last-col` (cf. 9, p. 36).

There is a particular case: for the delimiters `(`, `[` `\{`[54] and `\lbrace`, and the corresponding closing delimiters, the prefixes `\left` et `\right` are optional.[55]

Here is an example with a left delimiter `\{` in a `{NiceTabular}` (remark the compatibility with the key `t`).

```
We define $f$ with\quad
\begin{NiceTabular}[t]{\{ll}
$f(x) = 0$ & if $x$ is non positive \\
$f(x) = 1-e^x$ & if $x$ is positive
```

On définit f par $\begin{cases} f(x) = 0 & \text{if } x \text{ is non positive} \\ f(x) = 1 - e^x & \text{if } x \text{ is positive} \end{cases}$

When there are two successive delimiters (necessarily a closing one following by an opening one for another submatrix), a space equal to `\enskip` is automatically inserted.

```
$\begin{pNiceArray}{(c)(c)(c)}
a_{11} & a_{12}                                          & a_{13} \\
a_{21} & \displaystyle \int_0^1\dfrac{1}{x^2+1}\,dx & a_{23} \\
a_{31} & a_{32}                                          & a_{33}
\end{pNiceArray}$
```

$$\left(\begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix}\left(\int_0^1 \begin{matrix} a_{12} \\ \dfrac{1}{x^2+1}\,dx \\ a_{32} \end{matrix} \right)\begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix}\right)$$

For more complex constructions, in particular with delimiters spanning only a *subset* of the rows of the array, one should consider the command `\SubMatrix` available in the `\CodeAfter` (and the `\CodeBefore`). See the section 12.2, p. 48.

12 The \CodeAfter

The option `code-after` may be used to give some code that will be executed *after* the construction of the matrix.[56]

For the legibility of the code, an alternative syntax is provided: it's possible to give the instructions of the `code-after` at the end of the environment, after the keyword `\CodeAfter` (the key `code-after` is of course mandatory in the `\AutoNiceMatrix` and the similar commands). Although `\CodeAfter` is a keyword, it takes in an optional argument (between square brackets).[57]

The experienced users may, for instance, use the PGF/TikZ nodes created by nicematrix in the `\CodeAfter`. These nodes are described further in the section 15, p. 64.

The `\CodeAfter` provides several special commands: `\line`, `\SubMatrix`, `\OverBrace`, `\UnderBrace` and `\TikzEveryCell`. We will now present these commands.

[54] For the braces, the protection by a backslash is mandatory (that's why we have written `\{`).

[55] For the delimiters `[` and `]`, the prefixes remain mandatory when there is a conflict of notation with the square brackets for the options of some descriptors of columns.

[56] There is also a key `code-before` described in the section 6.2, p. 25.

[57] Here are the keys accepted in that argument: `delimiters/color`, `rules` and its sub-keys, `sub-matrix` (linked to the command `\SubMatrix`) and its sub-keys and `xdots` (for the command `\line`) and its sub-keys.

Remark: Since version 7.11a of nicematrix, that `\CodeAfter` is composed in mathematical mode and, therefore, the spurious spaces are deleted after a command such as `\SubMatrix`.

New 7.13

The key `overlap` controls whether the graphical elements drawn in the `\CodeAfter` are taken into account for the computation of the bounding box of the environment (`{NiceTabular}`, `{Nicematrix}`, etc.).
When the key `overlap` is set on `true`, those elements are *not* taken into account and they may overlap from the bounding box and be overwritten on the rest of the LaTeX document. On the opposite, when the key `overlap` is set on `false`, there is no overlaping because the elements added by the `\CodeAfter` are taken into account for the determination of the bounding box.
The initial value of the parameter `overlap` is `true` (and not `false`).

Remark: For technical reasons, the tuning `overlap=false` is *not* available in the historical way of compilation `latex-dvips-ps2pdf` (or Adobe Distiller).

12.1 The command `\line` in the `\CodeAfter`

The command `\line` draws directly dotted lines between cells or blocks. The mechanism for those dotted lines is the same as the mechanism used for the dotted leaders created by the commands `\Cdots`, `\Ldots`, etc. (cf. 10, p. 38). The command `\lines` takes in two arguments for the cells or blocks to link. Both arguments may be:

- a specification of cell of the form i-j where is the number of the row and j is the number of the column;
- the name of a block (created by the command `\Block` with the key `name` of that command).

The options available for the customisation of the dotted lines created by `\Cdots`, `\Vdots`, etc. are also available for this command (cf. 10.5, p. 42).

This command may be used, for example, to draw a dotted line between two adjacent cells.

```
$\begin{pNiceMatrix}[xdots/shorten += 0.3 em]
  I       & 0       & \Cdots  &0      \\
  0       & I       & \Ddots  &\Vdots\\
  \Vdots  &\Ddots   & I       &0      \\
  0       &\Cdots   & 0       &I
\CodeAfter \line{2-2}{3-3}
\end{pNiceMatrix}$
```

$$\begin{pmatrix} I & 0 & \cdots & 0 \\ & \ddots & & \vdots \\ 0 & I & \ddots & \\ \vdots & \ddots & I & 0 \\ 0 & \cdots & 0 & I \end{pmatrix}$$

It can also be used to draw a diagonal line not parallel to the other diagonal lines (by default, the dotted lines drawn by `\Ddots` are "parallelized": cf. 17.1, p. 71).

```
$\begin{bNiceMatrix}
  1      & \Cdots &   & 1      & 2      & \Cdots          & 2      \\
  0      & \Ddots &   & \Vdots & \Vdots & \hspace*{2.5cm} & \Vdots \\
  \Vdots & \Ddots &   &        &        &                 &        \\
  0      & \Cdots & 0 & 1      & 2      & \Cdots          & 2
\CodeAfter \line[shorten=6pt]{1-5}{4-7}
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 1 & \cdots & 1 & 2 & \cdots & 2 \\ & \ddots & \vdots & \vdots & \iddots & \vdots \\ 0 & & & & & \\ \vdots & \iddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 1 & 2 & \cdots & 2 \end{bmatrix}$$

12.2 The command \SubMatrix in the \CodeAfter (and the \CodeBefore)

The command `\SubMatrix` provides a way to put delimiters on a portion of the array considered as a submatrix. The command `\SubMatrix` takes in five arguments:

- the first argument is the left delimiter, which may be any extensible delimiter provided by LaTeX : `(`, `[`, `\{`, `\langle`, `\lgroup`, `\lfloor`, etc. but also the null delimiter `.`;
- the second argument is the upper-left corner of the submatrix with the syntax i-j where i the number of row and j the number of column (the special word `last` is allowed);
- the third argument is the lower-right corner with the same syntax;
- the fourth argument is the right delimiter;
- the last argument, which is optional, is a list of *key=value* pairs.[58]

One should remark that the command `\SubMatrix` draws the delimiters *after* the construction of the array: no space is inserted by the command `\SubMatrix` itself. That's why, in the following example, we have used the key `margin` and you have added by hand some space between the third and fourth column with `@{\hspace{1.5em}}` in the preamble of the array.

```
$\begin{NiceArray}{ccc@{\hspace{1.5em}}c}[cell-space-limits += 2pt,margin]
   1           & 1           & 1           & x \\
  \dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} & y \\
   1           & 2           & 3           & z
\CodeAfter
  \SubMatrix({1-1}{3-3})
  \SubMatrix({1-4}{3-4})
\end{NiceArray}$
```

$$\begin{pmatrix} 1 & 1 & 1 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Eventually, in this example, it would probably have been easier to put the delimiters directly in the preamble of `{NiceArray}` (cf. 11, p. 45) with the following construction.

```
$\begin{NiceArray}{(ccc)(c)}[cell-space-limits += 2pt]
   1           & 1           & 1           & x \\
  \dfrac{1}{4} & \dfrac{1}{2} & \dfrac{1}{4} & y \\
   1           & 2           & 3           & z
\end{NiceArray}$
```

$$\begin{pmatrix} 1 & 1 & 1 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

In fact, the command `\SubMatrix` also takes in two optional arguments specified by the traditional symbols `^` and `_` for material in superscript and subscript (but no space is reserved for that material and that's why, in the following example, you have used the key `right-margin`).

```
$\begin{bNiceMatrix}[right-margin=1em]
  1 & 1 & 1 \\
  1 & a & b \\
  1 & c & d
\CodeAfter
  \SubMatrix[{2-2}{3-3}]^{T}
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & \begin{bmatrix} a & b \\ c & d \end{bmatrix}^T \\ 1 & \end{bmatrix}$$

The options of the command `\SubMatrix` are as follows:

- `left-xshift` and `right-xshift` shift horizontally the delimiters (there exists also the key `xshift` which fixes both parameters);

[58] There is no optional argument between square brackets in first position because a square bracket just after `\SubMatrix` must be interpreted as the first (mandatory) argument of the command `\SubMatrix`: that bracket is the left delimiter of the sub-matrix to construct (eg.: `\SubMatrix[{2-2}{4-7}]`).

- `extra-height` adds a quantity to the total height of the delimiters (height `\ht` + depth `\dp`);
- `delimiters/color` fixes the color of the delimiters (also available in `\NiceMatrixOptions`, in the environments with delimiters and as option of the keyword `\CodeAfter`);
- `slim` is a boolean key: when that key is in force, the horizontal position of the delimiters is computed by using only the contents of the cells of the submatrix whereas, in the general case, the position is computed by taking into account the cells of the whole columns implied in the submatrix (see example below). ;
- `vlines` contents a list of numbers of vertical rules that will be drawn in the sub-matrix (if this key is used without value, all the vertical rules of the sub-matrix are drawn);
- `hlines` is similar to `vlines` but for the horizontal rules;
- `hvlines`, which must be used without value, draws all the vertical and horizontal rules;
- `code` insert code, especially TikZ code, after the construction of the submatrix. That key is detailed below.

One should remark that the keys add their rules after the construction of the main matrix: no space is added between the rows and the columns of the array for these rules.

All these keys are also available in `\NiceMatrixOptions`, at the level of the environments of nicematrix or as option of the command `\CodeAfter` with the prefix `sub-matrix` which means that their names are therefore `sub-matrix/left-xshift`, `sub-matrix/right-xshift`, `sub-matrix/xshift`, etc.

```
$\begin{NiceArray}{cc@{\hspace{5mm}}l}[cell-space-limits=2pt]
    &   & \frac{1}{2} \\
    &   & \frac{1}{4} \\[1mm]
  a & b & \frac{1}{2}a+\frac{1}{4}b \\
  c & d & \frac{1}{2}c+\frac{1}{4}d \\
\CodeAfter
  \SubMatrix({1-3}{2-3})
  \SubMatrix({3-1}{4-2})
  \SubMatrix({3-3}{4-3})
\end{NiceArray}$
```

$$\begin{pmatrix} \frac{1}{2} & \\ \frac{1}{4} & \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \frac{1}{2}a+\frac{1}{4}b \\ \frac{1}{2}c+\frac{1}{4}d \end{pmatrix}$$

Here is the same example with the key `slim` used for one of the submatrices.

```
$\begin{NiceArray}{cc@{\hspace{5mm}}l}[cell-space-limits=2pt]
    &   & \frac{1}{2} \\
    &   & \frac{1}{4} \\[1mm]
  a & b & \frac{1}{2}a+\frac{1}{4}b \\
  c & d & \frac{1}{2}c+\frac{1}{4}d \\
\CodeAfter
  \SubMatrix({1-3}{2-3})[slim]
  \SubMatrix({3-1}{4-2})
  \SubMatrix({3-3}{4-3})
\end{NiceArray}$
```

$$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{4} \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \frac{1}{2}a+\frac{1}{4}b \\ \frac{1}{2}c+\frac{1}{4}d \end{pmatrix}$$

There is also a key `name` which gives a name to the submatrix created by `\SubMatrix`. That name is used to create PGF/TikZ nodes: cf. 15.4, p. 69.

Despite its name, the command `\SubMatrix` may also be used within a `{NiceTabular}`. Here is an example (which uses `\bottomrule` and `\toprule` of booktabs).

```
\begin{NiceTabular}{@{}ll@{}}
\toprule
  Part A              & the first part \\
  \Block{2-1}{Part B} & a first sub-part \\
                      & a second sub-part \\
\bottomrule
\CodeAfter
  \SubMatrix{\lbrace}{2-2}{3-2}{.}
\end{NiceTabular}
```

Part A	the first part
Part B	a first sub-part
	a second sub-part

The command `\SubMatrix` is, in fact, also available in the `\CodeBefore`. By using `\SubMatrix` in the `\CodeBefore`, the delimiters drawn by those commands `\SubMatrix` are taken into account to limit the continuous dotted lines (drawn by `\Cdots`, `\Vdots`, etc.) which have an open extremity. For an example, see 18.11, p. 86.

Caution : The following functionality is fragile and does not work with `latex`–`dvips`–`ps2pdf`.

The key `code` of the command `\SubMatrix` allows the insertion of code after the construction of the submatrix. It's meant to be used to insert TikZ instructions because, in the TikZ instructions inserted by that code, the nodes of the form `i-j.`*anchor* or `i-|j` are interpreted with `i` and `j` as numbers of row and column *relative to the submatrix*.[59]

```
$\begin{NiceArray}{ccc@{}w{c}{5mm}@{}ccc}
    &   &   && -1 & 1  & 2  \\
    &   &   &&  0 & 3  & 4  \\
    &   &   &&  0 & 0  & 5  \\
  1 & 2 & 3 && -1 & 7  & 25 \\
  0 & 4 & 5 && 0  & 12 & 41 \\
  0 & 0 & 6 && 0  & 0  & 30
\CodeAfter
  \NewDocumentCommand{\MyDraw}{}{\tikz \draw [blue] (2-|1) -| (3-|2) -| (4-|3) ;}
  \SubMatrix({1-5}{3-7})[code = \MyDraw ]
  \SubMatrix({4-1}{6-3})[code = \MyDraw ]
  \SubMatrix({4-5}{6-7})[code = \MyDraw ]
\end{NiceArray}$
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{pmatrix} \begin{pmatrix} -1 & 1 & 2 \\ 0 & 3 & 4 \\ 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} -1 & 7 & 25 \\ 0 & 12 & 41 \\ 0 & 0 & 30 \end{pmatrix}$$

As we see, the drawing done by our command `\MyDraw` is *relative* to the submatrix to which it is applied.

12.3 The commands `\OverBrace` and `\UnderBrace` in the `\CodeAfter`

The commands `\OverBrace` and `\UnderBrace` provide a way to put horizontal braces on a part of the array. These commands take in three arguments:

- the first argument is the upper-left corner of the submatrix with the syntax i-j where i the number of row and j the number of column;
- the second argument is the lower-right corner with the same syntax;

[59] Be careful: the syntax `j|-i` is *not* allowed.

- the third argument is the label of the brace that will be put by nicematrix (with PGF) above the brace (for the command `\OverBrace`) or under the brace (for `\UnderBrace`).

  It's possible to use the command `\\` in this third argument in order to format the label on several lines.

By default there isn't any vertical space reserved for the brace and its label. In order to have one, it's necessary to use the tuning `overlap=false`.

```
Here is an example:

$\begin{pNiceMatrix}[overlap=false]
1  & 2  & 3  & 4  & 5  & 6  \\
11 & 12 & 13 & 14 & 15 & 16 \\
\CodeAfter
  \OverBrace{1-1}{2-3}{A}
  \OverBrace{1-4}{2-6}{B}
\end{pNiceMatrix}$
```

Here is an example:

$$\overbrace{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 11 & 12 & 13 & 14 & 15 & 16 \end{pmatrix}}^{\begin{matrix} A & & & B & & \end{matrix}}$$

In fact, the commands `\OverBrace` and `\UnderBrace` take in an optional argument (in first position and between square brackets) for a list of *key=value* pairs. The available keys are:

- `left-shorten` and `right-shorten` which do not take in value; when the key `left-shorten` is used, the abscissa of the left extremity of the brace is computed with the contents of the cells of the involved sub-array, otherwise, the position of the potential vertical rule is used (idem for `right-shorten`).
- `shorten`, which is the conjunction of the keys `left-shorten` and `right-shorten`;
- `yshift`, which shifts vertically the brace (and its label);
- `color`, which sets the color of the brace (and its label).

```
$\begin{pNiceMatrix}
  1  & 2  & 3  & 4  & 5  & 6  \\
  11 & 12 & 13 & 14 & 15 & 16 \\
\CodeAfter
  \OverBrace[shorten,yshift=3pt]{1-1}{2-3}{A}
  \OverBrace[shorten,yshift=3pt]{1-4}{2-6}{B}
\end{pNiceMatrix}$
```

$$\overbrace{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 11 & 12 & 13 & 14 & 15 & 16 \end{pmatrix}}^{\begin{matrix} A & & & B & & \end{matrix}}$$

▷ The commands `\OverBrace` and `\UnderBrace` use the standard braces of the mathematical mode of LaTeX on which they are based. However, nicematrix provides also a command `\Hbrace` which should be used directly in the tabular and which draws braces of TikZ (cf. 10.7, p. 44).

12.4 The command `\TikzEveryCell`

The command `\TikzEveryCell` executes with TikZ the rectangular path corresponding to each cell of the tabular with parameters of TikZ the argument of `\TikzEveryCell`. That argument must be a list of *key=value* pairs which may be applied to a TikZ path. In fact, the command applies to each cell of the tabular, except those in the exterior rows and columns (cf. 9, p. 36) and those in the empty corners (when the key `corners` is used: cf. 5.3.3, p. 17). It applies in fact to each block (excepted those with the key `transparent`) and does not apply to the individual cells located within these blocks.

In fact, in the list of keys provided as argument of `\TikzEveryCell`, it's possible to put a key `offset`. That key is *not* provided by TikZ but by nicematrix. It will narrow the rectangular frame corresponding to the block by a margin (horizontally and vertically) equal to the value (of that key `offset`). That new frame, a bit narrower, will be executed by TikZ with options which are the other keys in the argument of `\TikzEveryCell`.

```
\renewcommand{\arraystretch}{1.3}
\begin{NiceTabular}{ccc}[corners]
  & \Block{1-2}{columns} \\
  \Block{2-1}{rows}
  & cell 1 1 & cell 1 2 \\
  & cell 2 1 & cell 2 2
\CodeAfter
  \TikzEveryCell{offset=1pt,draw}
\end{NiceTabular}
```

	columns	
rows	cell 1 1	cell 1 2
	cell 2 1	cell 2 2

The command `\TikzEveryCell` is, in fact, also available in the `\CodeBefore`.

```
\renewcommand{\arraystretch}{1.3}
\begin{NiceTabular}{ccc}[corners]
\CodeBefore
  \TikzEveryCell
    {
      offset=1pt,
      fill=blue!15,
      rounded corners=2pt
    }
\Body
  & \Block{1-2}{columns} \\
  \Block{2-1}{rows}
  & cell 1 1 & cell 1 2 \\
  & cell 2 1 & cell 2 2
\end{NiceTabular}
```

	columns	
rows	cell 1 1	cell 1 2
	cell 2 1	cell 2 2

The command `\TikzEveryCell` has two optional keys, available between square brackets.

- with the key `empty`, the command only acts on the empty cells (the exact definition of an "empty cell" is given in part 17.2, p. 72);
- with the key `not-empty`, the command only acts on the not-empty cells.

```
\renewcommand{\arraystretch}{1.4}
\begin{NiceTabular}{cccccc}[hvlines]
  P & O & U & R & V & U \\
  O &   &   & E & I &   \\
  M & O & R & F & A & L \\
  E & T & A & L &   & E \\
  L & A & S & E & R & S \\
  O &   & E & X & I & T
\CodeAfter
  \TikzEveryCell[empty]{fill=gray,draw}
\end{NiceTabular}
```

P	O	U	R	V	U
O			E	I	
M	O	R	F	A	L
E	T	A	L		E
L	A	S	E	R	S
O		E	X	I	T

```
\begin{NiceTabular}{ccccc}
1 & 2 & 3 & 4 &   \\
1 &   &   & 4 & 5 \\
1 &   &   &   & 5 \\
1 & 2 &   & 4 & 5 \\
1 & 2 & 3 & 4 & 5 \\
\CodeAfter
  \TikzEveryCell[not-empty]{draw}
\end{NiceTabular}
```

1	2	3	4	
1			4	5
1				5
1	2		4	5
1	2	3	4	5

13 Captions and notes in the tabulars

13.1 Caption of a tabular

The environment `{NiceTabular}` provides the keys `caption`, `short-caption` and `label` which may be used when the tabular is inserted in a floating environment (typically the environment `{table}`).

With the key `caption`, the caption, when it is long, is wrapped at the width of the tabular (except the potential exterior columns specified by `first-col` and `last-col`: cf. 9, p. 36), without the use of the package threeparttable or the package floatrow.

By default, the caption is composed below the tabular. With the key `caption-above`, available in `\NiceMatrixOptions`, the caption will be composed above the tabular (that key is not available for an individual environment).

The key `short-caption` corresponds to the optional argument of the classical command `\caption` and the key `label` corresponds, of course, to the command `\label`.

See table 1, p. 55, for an example of use the keys `caption` and `label`.

These functionalities are compatible with the extension caption. For example, it's possible to use the tuning `\captionsetup{justification=raggedright,singlelinecheck=false}` in order to have left-align captions, even when the caption is shorter than the tabular.

13.2 The footnotes

The package nicematrix allows, by using footnote or footnotehyper, the extraction of the notes inserted by `\footnote` in the environments of nicematrix and their composition in the foot of the page with the other notes of the document.

If nicematrix is loaded with the option `footnote` (with `\usepackage[footnote]{nicematrix}` or with `\PassOptionsToPackage`), the package footnote is loaded (if it is not yet loaded) and it is used to extract the footnotes.

If nicematrix is loaded with the option `footnotehyper`, the package footnotehyper is loaded (if it is not yet loaded) ant it is used to extract footnotes.

Caution: The packages footnote and footnotehyper are incompatible. The package footnotehyper is the successor of the package footnote and should be used preferently. The package footnote has some drawbacks, in particular: it must be loaded after the package xcolor and it is not perfectly compatible with hyperref.

13.3 The notes of tabular

The package nicematrix also provides a command `\tabularnote` which gives the ability to specify notes that will be composed at the end of the array with a width of text equal to the width of the array (except the potential exterior columns specified by `first-col` and `last-col`: cf. 9, p. 36). With no surprise, that command is available only in the environments `{NiceTabular}`, `{NiceTabular*}` and `{NiceTabularX}`.
In fact, this command is available only if the extension enumitem has been loaded (before or after nicematrix). Indeed, the notes are composed at the end of the array with a type of list provided by the package enumitem. If enumitem has not been loaded, an error will be raised.

```
\begin{NiceTabular}{@{}llr@{}}
\toprule \RowStyle{\bfseries}
Last name & First name & Birth day \\
\midrule
Achard\tabularnote{Achard is an old family of the Poitou.}
& Jacques & June 5, 2005 \\
Lefebvre\tabularnote{The name Lefebvre is an alteration of the name Lefebure.}
& Mathilde & January 23, 1975 \\
```

```
Vanesse & Stephany & October 30, 1994 \\
Dupont & Chantal & January 15, 1998 \\
\bottomrule
\end{NiceTabular}
```



Last name	First name	Birth day
Achard[a]	Jacques	June 5, 2005
Lefebvre[b]	Mathilde	January 23, 1975
Vanesse	Stephany	October 30, 1994
Dupont	Chantal	January 15, 1998

[a] Achard is an old family of the Poitou.
[b] The name Lefebvre is an alteration of the name Lefebure.

- If you have several successive commands `\tabularnote{...}` *with no space at all between them*, the labels of the corresponding notes are composed together, separated by commas (this is similar to the option `multiple` of footmisc for the footnotes).
- If a command `\tabularnote{...}` is exactly at the end of a cell (with no space at all after) and if the alignment mode of the column is `c` or `r`, the label of the note is composed in an overlapping position (towards the right). This structure may provide a better alignment of the cells of a given column.
- If the key `notes/para` is used, the notes are composed at the end of the array in a single paragraph (as with the key `para` of threeparttable).
- There is a key `tabularnote` which provides a way to insert some text in the zone of the notes before the numbered tabular notes.

  An alternative syntax is available with the environment `{TabularNote}`. That environment should be used at the end of the environment `{NiceTabular}` (but *before* a potential instruction `\CodeAfter`).
- If the package booktabs has been loaded (before or after nicematrix), the key `notes/bottomrule` draws a `\bottomrule` of booktabs *after* the notes.
- The command `\tabularnote` may be used *before* the environment of nicematrix. Thus, it's possible to use it on the title inserted by `\caption` in an environment `{table}` of LaTeX (or in a command `\captionof` of the package caption). It's also possible, as expected, to use the command `\tabularnote` in the caption provided by the *key* `caption` of the environment `{NiceTabular}`.
- If several commands `\tabularnote` are used in a tabular with the same argument, only one note is inserted at the end of the tabular (but all the labels are composed, of course). It's possible to control that feature with the key `notes/detect-duplicates`.[60]
- It's possible to create a reference to a tabular note created by `\tabularnote` (with the usual command `\label` used after the `\tabularnote`).
- The command `\tabularnote` has an optional argument (between square brackets) to change the symbol of the reference of the note.

  Example: `\tabularnote[$\star$]{A footnote...}`

[60] For technical reasons, the final user is not allowed to put several commands `\tabularnote` with exactly the same argument in the caption of the tabular. It's not a real restriction...

For an illustration of some of those remarks, see table 1, p. 55. This table has been composed with the following code (the package caption has been loaded in this document).

```
\begin{table}[hbt]
\centering
\NiceMatrixOptions{caption-above}
\begin{NiceTabular}{@{}llc@{}
  [
    caption = A tabular whose caption has been specified by the key
      \texttt{caption}\tabularnote[$\star$]{It's possible to put a tabular
      note in the caption.} ,
    label = t:tabularnote ,
    tabularnote = Some text before the notes. ,
    notes/bottomrule
  ]
\toprule
  Last name & First name & Length of life \\
\midrule
  Churchill & Wiston &  91\\
  Nightingale\tabularnote{Considered as the first nurse of history}%
  \tabularnote{Nicknamed ``the Lady with the Lamp''.}
  & Florence\tabularnote{This note is shared by two references.} & 90 \\
  Schoelcher & Victor & 89\tabularnote{The label of the note is overlapping.}\\
  Touchet & Marie\tabularnote{This note is shared by two references.} & 89 \\
  Wallis & John & 87 \\
\bottomrule
\end{NiceTabular}
\end{table}
```



Table 1: A tabular whose caption has been specified by the key `caption`*

Last name	First name	Length of life
Churchill	Wiston	91
Nightingale^{a,b}	Florence^{c}	90
Schoelcher	Victor	89^{d}
Touchet	Marie^{c}	89
Wallis	John	87

Some text before the notes.

* It's possible to put a tabular note in the caption

^{a} Considered as the first nurse of history.

^{b} Nicknamed “the Lady with the Lamp”.

^{c} This note is shared by two references.

^{d} The label of the note is overlapping.

13.4 Customisation of the tabular notes

The tabular notes can be customized with a set of keys available in `\NiceMatrixOptions`. The name of these keys is prefixed by `notes`.f

- `notes/para`
- `notes/bottomrule`
- `notes/style`
- `notes/label-in-tabular`
- `notes/label-in-list`

- notes/enumitem-keys
- notes/enumitem-keys-para
- notes/code-before
- notes/no-print

For sake of commodity, it is also possible to set these keys in \NiceMatrixOptions via a key notes which takes in as value a list of pairs *key=value* where the name of the keys need no longer be prefixed by notes:

```
\NiceMatrixOptions
  {
    notes =
     {
       bottomrule ,
       style = ... ,
       label-in-tabular = ... ,
       enumitem-keys =
        {
          labelsep = ... ,
          align = ... ,
          ...
        }
     }
  }
```

We detail these keys.

- The key notes/para requires the composition of the notes (at the end of the tabular) in a single paragraph.

  Initial value: false

  That key is also available within a given environment.

- The key notes/bottomrule adds a \bottomrule of booktabs *after* the notes. Of course, that rule is drawn only if there actually are notes in the tabular. The package booktabs must have been loaded (before or after the package nicematrix). If it is not, an error is raised.

  Initial value: false

  That key is also available within a given environment.

- The key notes/style is a command whose argument is specified by #1 and which gives the style of numerotation of the notes. That style will be used by \ref when referencing a tabular note marked with a command \label. The labels formatted by that style are used, separated by commas, when the user puts several consecutive commands \tabularnote. The marker #1 is meant to be the name of a LaTeX counter.

  Initial value: \textit{\alph{#1}}

  Another possible value should be a mere \arabic{#1}

- The key notes/label-in-tabular is a command whose argument is specified by #1 which is used when formatting the label of a note in the tabular. Internally, this number of note has already been formatted by notes/style before sent to that command.

  Initial value: ^{#1}

  In French, it's a tradition of putting a small space before the label of note. That tuning could be achieved by the following code:

  ```
  \NiceMatrixOptions{notes/label-in-tabular = \,\textsuperscript{~#1}}
  ```

- The key `notes/label-in-list` is a command whose argument is specified by `#1` which is used when formatting the label in the list of notes at the end of the tabular. Internally, this number of note has already been formatted by `notes/style` before sent to that command.

  Initial value: `\textsuperscript{#1}`

  In French, the labels of notes are not composed in upper position when composing the notes. Such behaviour could be achieved by:

  ```
  \NiceMatrixOptions{notes/label-in-list = ~#1.\nobreak\hspace{0.25em}}
  ```

  The command `\nobreak` is for the event that the option `para` is used.

- The notes are composed at the end of the tabular by using internally a style of list of enumitem. This style of list is defined as follows (with, of course, keys of enumitem):

  `noitemsep , leftmargin = * , align = left , labelsep = 0pt`

  The specification `align = left` in that style requires a composition of the label leftwards in the box affected to that label. With that tuning, the notes are composed flush left, which is pleasant when composing tabulars in the spirit of booktabs (see for example the table 1, p. 55).

  The key `notes/enumitem-keys` specifies a list of pairs *key=value* (following the specifications of enumitem) to customize that style of list (it uses internally the command `\setlist*` of enumitem).

- The key `notes/enumitem-keys-para` is similar to the previous one but corresponds to the type of list used when the option `para` is in force. Of course, when the option `para` is used, a list of type `inline` (as called by enumitem) is used and the pairs *key=value* should correspond to such a list of type `inline`.

  Initially, the style of list is defined by: `afterlabel = \nobreak, itemjoin = \quad`

- The key `notes/code-before`$^+$ is a token list inserted by nicematrix just before the composition of the notes at the end of the tabular.

  Initial value: *empty*

  For example, if one wishes to compose all the notes in gray and `\footnotesize`, he should use that key:

  ```
  \NiceMatrixOptions{notes/code-before = \footnotesize \color{gray}}
  ```

  It's also possible to add `\raggedright` or `\RaggedRight` in that key (`\RaggedRight` is a command of ragged2e).

  The key `nodes/code-before` has a normal syntax, but also an "additive syntax", which means that it's possible to write, for example `notes/code-before += \footnotesize`. With that syntax, the tokens are appended on the *right* of the current value of the parameter `notes/code-before`.

- The key `notes/detect-duplicates` activates the detection of the commands `\tabularnotes` with the same argument.

  Initial value: `true`

- When the key `notes/no-print` is in force, the notes are *not* composed by nicematrix under the tabular. However, the final user may use the command `\NiceTabularNotes` in order to insert these notes where he wishes (after the composition of the tabular).

  Initial value: `false`

For an example of customisation of the tabular notes, see 18.1, p. 74.

14 Other features

14.1 Trees

The key `draw-trees-in-col` takes in as argument a list of numbers of columns and, in each of those columns, will draw a tree in the style the trees drawn by the packages dirtree and dirtreex.

```
\usepackage{booktabs} % in the preamble

\renewcommand{\arraystretch}{1.3}
\begin{NiceTabular}{w{l}{2.5mm}w{l}{2.5mm}lr}[draw-trees-in-col=1-2]
\toprule
  Category &&&  \% \\
\midrule
  Animal   &&& 100 \\
  & Human   &&  50 \\
  && Man     &  20 \\
  && Woman   &  30 \\
  & Fox     &&  30 \\
  && Vixen   &  16 \\
  && Dog     &  14 \\
  & Chicken &&  20 \\
  && Cock    &   8 \\
  && Hen     &  12 \\
\bottomrule
\end{NiceTabular}
```



Category	%
Animal	100
├── Human	50
│ ├── Man	20
│ └── Woman	30
├── Fox	30
│ ├── Vixen	16
│ └── Dog	14
└── Chicken	20
├── Cock	8
└── Hen	12

Three keys `trees/line-width`, `trees/color` and `trees/rounded-corners` are provided for the customization of those rules (the prefix `trees` may be factorized).

```
\renewcommand{\arraystretch}{1.3}
\NiceMatrixOptions{trees= {line-width = 0.4pt, color = red , rounded-corners}}
\begin{NiceTabular}{w{l}{2.5mm}w{l}{2.5mm}lr}[draw-trees-in-col=1-2]
\toprule
  Category &&&  \% \\
\midrule
  Animal   &&& 100 \\
  & Human   &&  50 \\
  && Man     &  20 \\
  && Woman   &  30 \\
  & Fox     &&  30 \\
```

```
  && Vixen    &  16 \\
  && Dog      &  14 \\
  & Chicken &&  20 \\
  && Cock     &   8 \\
  && Hen      &  12 \\
\bottomrule
\end{NiceTabular}
```



Category	%
Animal	100
Human	50
Man	20
Woman	30
Fox	30
Vixen	16
Dog	14
Chicken	20
Cock	8
Hen	12

14.2 The key rounded-corners of the environments

The key `rounded-corners` that we will describe now has no direct link with the key `corners` (which is used to specify "empty corners") described in the part 5.3.3, p. 17. On the other side, it is similar with the key `rounded-corners` of the command `\Block` (cf. 4.1, 7).

The key `rounded-corners` specifies that the tabular should have rounded corners with a radius equal to the value of that key (the default value is 4 pt[61]). More precisely, that key has two effects that we describe now.

- All the commands for coloring the cells, columns and rows (in the `\CodeBefore` but also directly in the array) will respect those rounded corners.
- When the key `hvlines` is used, the exterior rules will be drawn with rounded corners.[62]

That key is available in all the environments and commands (e.g. `\pAutoNiceMatrix`) of nicematrix and also in the command `\NiceMatrixOptions`.

```
\begin{NiceTabular}
  [hvlines,rounded-corners]
  {ccc}
\CodeBefore
  \rowcolor{red!15}{1}
\Body
  Last name & First name & Profession \\
  Arvy & Jacques & Physicist \\
  Jalon & Amandine & Physicist
\end{NiceTabular}
```

Last name	First name	Profession
Arvy	Jacques	Physicist
Jalon	Amandine	Physicist



[61] This value is the initial value of the *rounded corners* of TikZ.

[62] Of course, when used in an environment with delimiters (`{pNiceMatrix}`, `{bNiceArray}`, etc.), the key `hvlines` does not draw the exterior rules.

14.3 The command \ShowCellNames

The command `\ShowCellNames`, which may be used in the `\CodeBefore` and in the `\CodeAfter` displays the name (with the form *i*-*j*) of each cell. It is provided as a tool which may be helpul during the conception of a table.

When it is used in the `\CodeBefore`, the names are drawn after the colored backgrounds (but before the contents of the cells).

```
\begin{NiceTabular}{ccc}[hvlines,cell-space-limits=3pt]
\CodeBefore
  \ShowCellNames
  \rowcolor{gray!15}{3}
\Body
  \Block{2-2}{} &           & test \\
                &           & blah blah \\
                & some text & nothing
\end{NiceTabular}
```

1-1	1-2	1-3
2-1	2-2	2-3
3-1	3-2	3-3

When used in the `\CodeAfter`, that command applies a semi-transparent white rectangle to fade the array (caution: some PDF readers don't support transparency) and, then, the names of the cells are drawn above.

```
\begin{NiceTabular}{ccc}[hvlines,cell-space-limits=3pt]
\CodeBefore
  \rowcolor{gray!15}{3}
\Body
  \Block{2-2}{} &           & test \\
                &           & blah blah \\
                & some text & nothing
\CodeAfter \ShowCellNames
\end{NiceTabular}
```

1-1	1-2	1-3
2-1	2-2	2-3
3-1	3-2	3-3

14.4 Use of the column type S of siunitx

If the package siunitx is loaded (before or after nicematrix), it's possible to use the `S` column type of siunitx in the environments of nicematrix.

```
$\begin{pNiceArray}{ScW{c}{1cm}c}[xdots/nullify,first-row]
  {C_1} & \Cdots &  & C_n \\
  2.3  & 0 & \Cdots & 0 \\
  12.4 & \Vdots & & \Vdots \\
  1.45 \\
  7.2  & 0 & \Cdots & 0
\end{pNiceArray}$
```

$$\begin{array}{cccc} C_1 & \cdots & & C_n \\ \left(\begin{array}{cc} 2.3 & 0 \\ 12.4 & \vdots \\ 1.45 & \vdots \\ 7.2 & 0 \end{array}\right. & \cdots & & \left.\begin{array}{c} 0 \\ \vdots \\ \vdots \\ 0 \end{array}\right) \end{array}$$

On the other hand, the `d` columns of the package dcolumn and the `n` columns of the package numprint are not supported by nicematrix.

14.5 Default column type in {NiceMatrix}

The environments without preamble (`{NiceMatrix}`, `{pNiceMatrix}`, `{bNiceMatrix}`, etc.) and the commande `\pAutoNiceMatrix` (and its variants) provide an option `columns-type` to specify the type of column which will be used (the initial value is, of course, `c`).
The keys `l` and `r` are shortcuts for `columns-type=l` and `columns-type=r`.

```
$\begin{bNiceMatrix}[r]
  \cos x & - \sin x \\
  \sin x & \cos x
\end{bNiceMatrix}$
```

$$\begin{bmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{bmatrix}$$

This syntax is similar to the syntax of the environment `{bmatrix*}`, fournie par l'exension mathtools.

The key `columns-type` is available in `\NiceMatrixOptions` but with the prefix `matrix`, which means that its name is, within `\NiceMatrixOptions` : `matrix/columns-type`.

14.6 The command \rotate

The package nicematrix provides a command `\rotate`. When used in the beginning of a cell, this command composes the contents of the cell after a rotation of 90° in the direct sens.
In the following command, we use that command in the `code-for-first-row`.[63]

```
\NiceMatrixOptions
 {
   code-for-first-row = \scriptstyle \rotate \text{image of },
   code-for-last-col = \scriptstyle
 }
$A = \begin{pNiceMatrix}[first-row,last-col=4]
  e_1 & e_2 & e_3       \\
  1   & 2   & 3   & e_1 \\
  4   & 5   & 6   & e_2 \\
  7   & 8   & 9   & e_3
\end{pNiceMatrix}$
```

$$A = \begin{array}{c} \begin{array}{ccc} \text{image of } e_1 & \text{image of } e_2 & \text{image of } e_3 \end{array} \\ \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{array}{c} e_1 \\ e_2 \\ e_3 \end{array} \end{array}$$

If the command `\rotate` is used in the "last row" (exterior to the matrix), the corresponding elements are aligned upwards as shown below.

```
\NiceMatrixOptions
  {
    code-for-last-row = \scriptstyle \rotate ,
    code-for-last-col = \scriptstyle
  }
$A = \begin{pNiceMatrix}[last-row=4,last-col=4]
  1   & 2   & 3   & e_1 \\
  4   & 5   & 6   & e_2 \\
  7   & 8   & 9   & e_3 \\
  \text{image of } e_1 & e_2 & e_3
\end{pNiceMatrix}$
```

$$A = \begin{array}{c} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \begin{array}{c} e_1 \\ e_2 \\ e_3 \end{array} \\ \begin{array}{ccc} \text{image of } e_1 & e_2 & e_3 \end{array} \end{array}$$

The command `\rotate` has a key `c`: `\rotate[c]` (spaces are deleted after `\rotate[c]`). When that key is used, the content is composed in a `\vcenter` and, therefore, in most cases, we will have a vertical alignment.

The command `\rotate` also supports a key called `-90`. With that key, the content is rotated by an angle of $-90°$ and not $90°$.

Caution: the command `\rotate` is designed to be used in a `\Block` or in columns of type `l`, `r`, `c`, `w` or `W`; if it is used in other column types (such as `p{...}`), the result will maybe differ from what is expected. In that case, it will be valuable to consider the command `\rotatebox` of the package graphicx (that package is loaded by nicematrix).

[63] It can also be used in `\RowStyle` (cf. 7, p. 32).

14.7 The key small

With the option `small`, the environments of the package nicematrix are composed in a way similar to the environment `{smallmatrix}` of the package amsmath (and the environments `{psmallmatrix}`, `{bsmallmatrix}`, etc. of the package mathtools).

```
$\begin{bNiceArray}{cccc|c}[small,
                           last-col ,
                           code-for-last-col = \scriptscriptstyle,
                           columns-width = 3mm ]
  1 & -2 & 3 & 4 & 5 \\
  0 & 3  & 2 & 1 & 2 & L_2 \gets 2 L_1 - L_2 \\
  0 & 1  & 1 & 2 & 3 & L_3 \gets L_1 + L_3
\end{bNiceArray}$
```

$$\left[\begin{array}{cccc|c} 1 & -2 & 3 & 4 & 5 \\ 0 & 3 & 2 & 1 & 2 \\ 0 & 1 & 1 & 2 & 3 \end{array}\right] \begin{array}{l} \\ L_2 \leftarrow 2L_1 - L_2 \\ L_3 \leftarrow L_1 + L_3 \end{array}$$

One should note that the environment `{NiceMatrix}` with the option `small` is not composed *exactly* as the environment `{smallmatrix}`. Indeed, all the environments of nicematrix are constructed upon `{array}` (of the package array) whereas the environment `{smallmatrix}` is constructed directly with an `\halign` of TeX.

In fact, the option `small` corresponds to the following tuning:

- the cells of the array are composed with `\scriptstyle`;
- `\arraystretch` is set to 0.47;
- `\arraycolsep` is set to 1.45 pt;
- the characteristics of the dotted lines are also modified.

When the key `small` is in force, some functionalities of nicematrix are no longer available: for example, it's no longer possible to put vertical delimiters directly in the preamble of an environment with preamble (cf. 11, p. 45).

14.8 `\AutoNiceMatrix` and the counters iRow and jCol

In the cells of the array, it's possible to use the LaTeX counters `iRow` and `jCol`[64] which represent the number of the current row and the number of the current column. We recall that the exterior "first row" (if it exists) has the number 0 and that the exterior "first column" (if it exists) has also the number 0. Of course, the user must not change the value of these counters `iRow` and `jCol` which are used internally by nicematrix.
In the `\CodeBefore` (cf. 6.2, p. 25) and in the `\CodeAfter` (cf. 12, p. 46), `iRow` represents the total number of rows (except the potential exterior rows: cf. 9, p. 36) and `jCol` represents the total number of columns (except the potential exterior columns).

```
$\begin{pNiceMatrix}% don't forget the %
    [first-row,
     first-col,
     code-for-first-row = \mathbf{\alph{jCol}} ,
     code-for-first-col = \mathbf{\arabic{iRow}} ]
  &   &    &    &   \\
  & 1 & 2  & 3  & 4 \\
  & 5 & 6  & 7  & 8 \\
  & 9 & 10 & 11 & 12
\end{pNiceMatrix}$
```

$$\begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \end{array} \\ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix} \end{array}$$

[64] These counters are actual LaTeX counters: the underlying TeX counters are `\c@iRow` and `\c@jCol`

If LaTeX counters called `iRow` and `jCol` are defined in the document by packages other than nicematrix (or by the final user), they are shadowed in the environments of nicematrix.

The package nicematrix also provides commands in order to compose automatically matrices from a general pattern. These commands are `\AutoNiceMatrix`, `\pAutoNiceMatrix`, `\bAutoNiceMatrix`, `\vAutoNiceMatrix`, `\VAutoNiceMatrix` et `\BAutoNiceMatrix`.
These commands take in two mandatory arguments. The first is the format of the matrix, with the syntax *n*-*p* where *n* is the number of rows and *p* the number of columns. The second argument is the pattern (it's a list of tokens which are inserted in each cell of the constructed matrix).

```
$C = \pAutoNiceMatrix{3-3}{C_{\arabic{iRow},\arabic{jCol}}}$
```

$$C = \begin{pmatrix} C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

There exists also `\AutoNiceArrayWithDelims` similar to `{NiceArrayWithDelims}`.

14.9 The key light-syntax

The option `light-syntax` (inpired by the package spalign) allows the user to compose the arrays with a lighter syntax, which gives a better legibility of the TeX source.
When this option is used, one should use the semicolon for the end of a row and spaces or tabulations to separate the columns. However, as usual in the TeX world, the spaces after a control sequence are discarded and the elements between curly braces are considered as a whole. These commands take in also an optional argument (for the standard options of nicematrix) in the first position and also another in the last position (in particular, it's possible to use the keys `code-before` and `code-after` in these arguments).

```
$\begin{bNiceMatrix}[light-syntax,first-row,first-col]
  {} a             b                 ;
  a  2\cos a       {\cos a + \cos b} ;
  b \cos a+\cos b  { 2 \cos b }
\end{bNiceMatrix}$
```

$$\begin{array}{c} \\ a \\ b \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{bmatrix} 2\cos a & \cos a + \cos b \\ \cos a + \cos b & 2\cos b \end{bmatrix} \end{array}$$

It's possible to change the character used to mark the end of rows with the option `end-of-row`. As said before, the initial value is a semicolon.

When the option `light-syntax` is used, it is not possible to put verbatim material (for example with the command `\verb`) in the cells of the array.[65]

The key `light-syntax-expanded` has the same behaviour as the key `light-syntax` but the body of the environment is expanded (in the TeX sense[66]) before being split in rows (but after the extraction of a potential `\CodeAfter`).

14.10 Color of the delimiters

For the environments with delimiters (`{pNiceArray}`, `{pNiceMatrix}`, etc.), it's possible to change the color of the delimiters with the key `delimiters/color`.

```
$\begin{bNiceMatrix}[delimiters/color=red]
  1 & 2 \\
  3 & 4
\end{bNiceMatrix}$
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

This color also applies to the delimiters drawn by the command `\SubMatrix` (cf. 12.2, p. 48) and for the delimiters directly specified in the preamble of the environments with preamble (cf. 11, p. 45).

[65] The reason is that, when the option `light-syntax` is used, the whole content of the environment is loaded as a TeX argument to be analyzed. The environment doesn't behave in that case as a standard environment of LaTeX which only put TeX commands before and after the content.

[66] More precisely, it's an expansion of type `e` of L3.

14.11 The environment {NiceArrayWithDelims}

In fact, the environment {pNiceArray} and its variants are based upon a more general environment, called {NiceArrayWithDelims}. The first two mandatory arguments of this environment are the left and right delimiters used in the construction of the matrix.
It's possible to use {NiceArrayWithDelims} if we want to use atypical or asymmetrical delimiters.

```
$\begin{NiceArrayWithDelims}
   {\downarrow}{\uparrow}{ccc}[margin]
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
\end{NiceArrayWithDelims}$
```

$$\begin{NiceArrayWithDelims}{\downarrow}{\uparrow}{ccc}[margin] 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{NiceArrayWithDelims}$$

14.12 The command \OnlyMainNiceMatrix

The command \OnlyMainNiceMatrix executes its argument only when it is in the main part of the array, that is to say when it is not in one of the exterior rows or one of the exterior columns. If it is used outside an environment of nicematrix, that command is no-op. That command is available in all the environments of nicematrix (and might have been called \OnlyMainNiceTabular).
For examples of use, see tex.stackexchange.com/questions/488566 and tex.stackexchange.com/questions/73412.

15 Explicit use of TikZ with nicematrix

15.1 The nodes corresponding to the contents of the cells

The package nicematrix creates a PGF/TikZ node[67] for the **content** of each (non-empty) cell of the array. These nodes, which, by default, are invisible in the PDF, are used to draw the dotted lines between the cells of the matrix (inter alia).

By default, those nodes are not available in the \CodeBefore (in order to speed up compilations). In order to be able to use those nodes in the \CodeBefore, one must use the key create-cell-nodes of the keyword \CodeBefore.

Caution : By default, no node is created in an empty cell.

However, it's possible to impose the creation of a node with the command \NotEmpty. [68]

The creation of those nodes needs time and memory. It's possible to desactive ponctually the creation of those nodes with the key no-cell-nodes in order to speed up the compilations. But be careful: those nodes are used internally but nicematrix for several features. One should use the key no-cell-nodes only when those feateures are not used. Among these are the key corners and the commands for the continuous dotted lines (\Cdots, etc.).

The nodes of a document must have distinct names. That's why the names of the nodes created by nicematrix contains the number of the current environment. Indeed, the environments of nicematrix are numbered by a internal global counter.

In the environment with the number n, the node of the row i and column j has for name nm-n-i-j.

The command \NiceMatrixLastEnv provides the number of the last environment of nicematrix (for LaTeX, it's a "fully expandable" command and not a counter).

[67] We recall that TikZ is a layer over PGF. The extension nicematrix loads PGF but does not load TikZ. We speak of PGF/TikZ nodes to emphase the fact that the PGF nodes created by nicematrix may be used with PGF but also with TikZ. The final user will probably prefer to use TikZ rather than PGF.

[68] One should note that, with that command, the cell is considered as non-empty, which has consequencies for the continuous dotted lines (cf. 10, p. 38) and the computation of the "corners" (cf. 5.3.3, p. 17).

However, it's advisable to use instead the key `name`[69]. This key gives a name to the current environment. When the environment has a name, the nodes are accessible with the name "*name-i-j*" where *name* is the name given to the array and i and j the numbers of row and column. It's possible to use these nodes with PGF but the final user will probably prefer to use TikZ (which is a convenient layer upon PGF). However, one should remind that nicematrix doesn't load TikZ by default. In the following examples, we assume that TikZ has been loaded.

```
$\begin{pNiceMatrix}[name=mymatrix]
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
\end{pNiceMatrix}$
\tikz[remember picture,overlay]
     \draw (mymatrix-2-2) circle (2mm) ;
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & \textcircled{5} & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Don't forget the options `remember picture` and `overlay`.

In the `\CodeAfter`, the things are easier : one must refer to the nodes with the form i-j (we don't have to indicate the environment which is of course the current environment).

```
$\begin{pNiceMatrix}
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
\CodeAfter
  \tikz \draw (2-2) circle (2mm) ;
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & \textcircled{5} & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

The nodes of the last column (except the potential "last column" specified by `last-col`[70]) may also be indicated by i-`last`. Similarly, the nodes of the last row may be indicated by `last`-j.

In the following example, we have stroken all the nodes of the matrix.

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix}$$

Since those nodes are PGF nodes, one won't be surprised to learn that they are drawn by using a specific PGF style. That style is called `nicematrix/cell-node` and its definition in the source file `nicematrix.sty` is as follows:

```
\pgfset
  {
    nicematrix / cell-node /.style =
     {
       inner sep = 0 pt ,
       minimum width = 0 pt
     }
  }
```

The final user may modify that style by changing the values of the keys `text/rotate`, `inner xsep`, `inner ysep`, `inner sep`, `outer xsep`, `outer ysep`, `outer sep`, `minimum width`, `minimum height` and `minimum size`.

For an example of use, see part 18.12, p. 87.

[69] The value of the key `name` is *expanded* (in the TeX sens).

[70] For the exterior columns, cf. 9, p. 36.

15.1.1 The key pgf-node-code

For the experienced users, nicematrix provides the key `pgf-node-code` which corresponds to some PGF node that will be executed at the creation, by PGF, of the nodes corresponding to the cells of the array. More pricisely, the value given to the key `pgf-node-code` will be passed in the fifth argument of the command `\pgfnode`. That value should contain at least an instruction such as `\pgfusepath`, `\pgfusepathqstroke`, `\pgfusepathqfill`, etc.

15.1.2 The columns V of varwidth

When the extension varwidth is loaded, the columns of the type `V` defined by varwidth are supported by nicematrix. It may be interessant to notice that, for a cell of a column of type `V`, the PGF/TikZ node created by nicematrix for the content of that cell has a width adjusted to the content of the cell. This is in contrast to the case of the columns of type `p`, `m` or `b` for which the nodes have always a width equal to the width of the column. In the following example, the command `\lipsum` is provided by the eponymous package.

```
\begin{NiceTabular}{V{10cm}}
  \bfseries \large
  Titre \\
  \lipsum[1][1-4]
\CodeAfter
  \tikz \draw [rounded corners] (1-1) -| (last-|2) -- (last-|1) |- (1-1) ;
\end{NiceTabular}
```

Titre

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

We have used the nodes corresponding to the position of the potential rules, which are described below (cf. 15.3, p. 68).
Of course, there are dedicated packages to such constructions (for example tcolorbox).

15.2 The "medium nodes" and the "large nodes"

In fact, the package nicematrix can create "extra nodes": the "medium nodes" and the "large nodes". The first ones are created with the option `create-medium-nodes` and the second ones with the option `create-large-nodes`.[71]

These nodes are not used by nicematrix by default, and that's why they are not created by default.

The names of the "medium nodes" are constructed by adding the suffix "`-medium`" to the names of the "normal nodes". In the following example, we have underlined the "medium nodes". We consider that this example is self-explanatory.

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix}$$

The names of the "large nodes" are constructed by adding the suffix "`-large`" to the names of the "normal nodes". In the following example, we have underlined the "large nodes". We consider that this example is self-explanatory.[72]

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix}$$

[71] There is also an option `create-extra-nodes` which is an alias for the conjunction of `create-medium-nodes` and `create-large-nodes`.

[72] There is no "large nodes" created in the exterior rows and columns (for these rows and columns, cf. 9, p. 36).

The "large nodes" of the first column and last column may appear too small for some usage. That's why it's possible to use the options `left-margin` and `right-margin` to add space on both sides of the array and also space in the "large nodes" of the first column and last column. In the following example, we have used the options `left-margin` and `right-margin`.[73]

$$\left(\begin{array}{ccc} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{array}\right)$$

It's also possible to add more space on both sides of the array with the options `extra-left-margin` and `extra-right-margin`. These margins are not incorporated in the "large nodes". It's possible to fix both values with the option `extra-margin` and, in the following example, we use `extra-margin` with the value 3 pt.

$$\left(\begin{array}{ccc} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{array}\right)$$

Be careful : These nodes are reconstructed from the contents of the contents cells of the array. Usually, they do not correspond to the cells delimited by the rules (if we consider that these rules are drawn).

Here is an array composed with the following code:

```
\large
\begin{NiceTabular}{w{l}{2cm}ll}[hvlines]
  fraise & amande & abricot \\
  prune & pêche & poire  \\[1ex]
  noix & noisette & brugnon
\end{NiceTabular}
```

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Here, we have colored all the cells of the array with `\chessboardcolors`.

fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

Here are the "large nodes" of this array (without use of `margin` nor `extra-margin`).

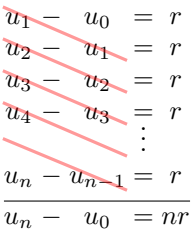
fraise	amande	abricot
prune	pêche	poire
noix	noisette	brugnon

The nodes we have described are not available by default in the `\CodeBefore` (cf. 6.2, p. 25).
It's possible to have these nodes available in the `\CodeBefore` by using the key `create-cell-nodes` of the keyword `\CodeBefore` (in that case, the nodes are created first before the construction of the array by using information written on the `aux` file and created a second time during the construction of the array itself; this mechanism is not activated by default for efficiency reasons).

Here is an example which uses these nodes in the `\CodeAfter`.

[73] The options `left-margin` and `right-margin` take dimensions as values but, if no value is given, the default value is used, which is `\arraycolsep` (by default: 5 pt). There is also an option `margin` to fix both `left-margin` and `right-margin` to the same value.

```
$\begin{NiceArray}{c@{\;}c@{\;}c@{\;}c@{\;}c}[create-medium-nodes]
   u_1 &-& u_0 &=& r      \\
   u_2 &-& u_1 &=& r      \\
   u_3 &-& u_2 &=& r      \\
   u_4 &-& u_3 &=& r      \\
   \phantom{u_5} & &  \phantom{u_4}    &\smash{\vdots} &       \\
   u_n &-& u_{n-1} &=& r \\[3pt]
   \hline
   u_n &-& u_0 &=& nr \\
\CodeAfter
   \tikz[very thick, red, opacity=0.4, name suffix = -medium]
     \draw (1-1.north west) -- (2-3.south east)
       (2-1.north west) -- (3-3.south east)
       (3-1.north west) -- (4-3.south east)
       (4-1.north west) -- (5-3.south east)
       (5-1.north west) -- (6-3.south east) ;
\end{NiceArray}$
```



$$\begin{array}{rcl} u_1 - u_0 & = & r \\ u_2 - u_1 & = & r \\ u_3 - u_2 & = & r \\ u_4 - u_3 & = & r \\ & \vdots & \\ u_n - u_{n-1} & = & r \\ \hline u_n - u_0 & = & nr \end{array}$$

15.3 The nodes which indicate the position of the rules

The package nicematrix creates a PGF/TikZ node merely called i (with the classical prefix) at the intersection of the horizontal rule of number i and the vertical rule of number i (more specifically the potential position of those rules because maybe there are not actually drawn). The last node has also an alias called `last`.

There are also nodes called i`.1`, i`.2`, i`.25`, i`.3`, i`.4`, i`.5`, i`.6`, i`.7`, i`.75`, i`.8` and i`.9` between the node i and the node $i+1$.

These nodes are available in the `\CodeBefore` and the `\CodeAfter`.

tulipe		lys
arum		violette mauve
muguet	dahlia	

If we use TikZ (we remind that nicematrix does not load TikZ by default, by only PGF, which is a sub-layer of TikZ), we can access, in the `\CodeAfter` but also in the `\CodeBefore`, to the intersection of the (potential) horizontal rule i and the (potential) vertical rule j with the syntax `(`i`-|`j`)`.

```
$\begin{NiceMatrix}
\CodeBefore
  \tikz \draw [fill=red!15] (7-|4) |- (8-|5) |- (9-|6) |- cycle ;
\Body
  1 \\
  1 & 1 \\
  1 & 2 &  1 \\
  1 & 3 &  3 &  1 \\
  1 & 4 &  6 &  4 &  1 \\
  1 & 5 & 10 & 10 &  5 &  1 \\
  1 & 6 & 15 & 20 & 15 &  6 &  1 \\
  1 & 7 & 21 & 35 & 35 & 21 &  7 & 1 \\
  1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1
\end{NiceMatrix}$
```



$$
\begin{array}{ccccccccc}
1 & & & & & & & & \\
1 & 1 & & & & & & & \\
1 & 2 & 1 & & & & & & \\
1 & 3 & 3 & 1 & & & & & \\
1 & 4 & 6 & 4 & 1 & & & & \\
1 & 5 & 10 & 10 & 5 & 1 & & & \\
1 & 6 & 15 & 20 & 15 & 6 & 1 & & \\
1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 & \\
1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1
\end{array}
$$

The "decimal" nodes (like $i.4$) may be used, for example to cross a row of a matrix (if TikZ is loaded).

```
$\begin{pNiceArray}{ccc|c}
  2 & 1 & 3 & 0 \\
  3 & 3 & 1 & 0 \\
  3 & 3 & 1 & 0
\CodeAfter
  \tikz \draw [red] (3.4-|1) -- (3.4-|last) ;
\end{pNiceArray}$
```

$$
\left(\begin{array}{ccc|c}
2 & 1 & 3 & 0 \\
3 & 3 & 1 & 0 \\
3 & 3 & 1 & 0
\end{array}\right)
$$



15.4 The nodes corresponding to the command \SubMatrix

The command `\SubMatrix` available in the `\CodeAfter` has been described in the section 12.2, p. 48.

If a command `\SubMatrix` has been used with the key `name` with an expression such as `name=`*`MyName`* three PGF/TikZ nodes are created with the names *`MyName`*`-left`, *`MyName`* and *`MyName`*`-right`.

The nodes *`MyName`*`-left` and *`MyName`*`-right` correspond to the delimiters left and right and the node *`MyName`* correspond to the submatrix itself.

In the following example, we have highlighted these nodes (the submatrix itself has been created with `\SubMatrix\{{2-2}{3-3}\}`).

$$
\left(\begin{array}{cccc}
121 & 23 & 345 & 345 \\
45 & \left\{\begin{array}{cc} 346 & 863 \\ 38458 & 34 \end{array}\right\} & & 444 \\
3462 & & & 294 \\
34 & 7 & 78 & 309
\end{array}\right)
$$

16 API for the developers

16.1 Public sets of keys

The extension nicematrix provides several public sets of keys:

- `nicematrix/environments` which applies to all the environments of nicematrix;
- `nicematrix/NiceMatrix` which applies specifically to the environment `{NiceMatrix}` and its variants with delimiters (`{pNiceMatrix}`, etc.);
- `nicematrix/NiceTabular` which applies specifically to the environements `{NiceTabular}`, `{NiceTabularX}` and `{NiceTabular*}`.

For example, if you wish to be able to fix the parameter `\arraystretch` with a key `arraystretch`, you only have to add the following instructions in the preamble of the LaTeX document.



```
\DeclareKeys[nicematrix/environments]{arraystretch.store=\arraystretch}
```

Then, it's possible to write, for example:

```
\begin{NiceTabular}{cc}[hvlines,arraystretch=2]
a & b \\
c & d
\end{NiceTabular}
```

a	b
c	d



16.2 \CodeBefore and \CodeAfter

The package nicematrix provides two variables which are internal but public[74]:

- \g_nicematrix_code_before_tl ;
- \g_nicematrix_code_after_tl.

These variables contain the code of what we have called the "code-before" (usually specified at the beginning of the environment with the syntax using the keywords \CodeBefore and \Body) and the "code-after" (usually specified at the end of the environment after the keyword \CodeAfter). The developer can use them to add code from a cell of the array (the affectation must be global, allowing to exit the cell, which is a TeX group).

One should remark that the use of \g_nicematrix_code_before_tl needs one compilation more (because the instructions are written on the aux file to be used during the next run).

Example : We want to write a command \crossbox to draw a cross in the current cell. This command will take in an optional argument between square brackets for a list of pairs *key-value* which will be given to TikZ before the drawing. It's possible to program such command \crossbox as follows, explicitly using the public variable \g_nicematrix_code_before_tl.
We use \g_nicematrix_code_before_tl instead of \g_nicematrix_code_after_tl because we want to draw the cross before the horizontal aud vertical rules (in order to have a perfect output).

```
\ExplSyntaxOn
\cs_new_protected:Nn \__pantigny_crossbox:nnn
  {
    \tikz \draw [ #3 ]
          ( #1 -| \inteval { #2 + 1 } ) -- ( \inteval { #1 + 1 } -| #2 )
          ( #1 -| #2 ) -- ( \inteval { #1 + 1 } -| \inteval { #2 + 1 } ) ;
  }

\NewDocumentCommand \crossbox { ! O { } }
  {
    \tl_gput_right:Ne \g_nicematrix_code_before_tl
      {
        \__pantigny_crossbox:nnn
          { \arabic { iRow } }
          { \arabic { jCol } }
          { \exp_not:n { #1 } }
      }
    \ignorespaces
  }
\ExplSyntaxOff
```



We have used the LaTeX counters iRow and jCol provided by nicematrix (cf. 14.8, p. 62).

Here is an example of use:

[74] According to the LaTeX3 conventions, each variable with name beginning with \g_nicematrix ou \l_nicematrix is public and each variable with name beginning with \g__nicematrix or \l__nicematrix is private.

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \arraycolor{gray!10}
\Body
  merlan & requin & cabillaud \\
  baleine & \crossbox[red] & morue \\
  mante & raie & poule
\end{NiceTabular}
```

merlan	requin	cabillaud
baleine		morue
mante	raie	poule

17 Technical remarks

First remark: the package underscore must be loaded before nicematrix. If it is loaded after, an error will be raised.

17.1 Diagonal dotted leaders

By default, all the diagonal dotted lines[75] of a same array are "parallelized". That means that the first diagonal line is drawn and, then, the other lines are drawn parallel to the first one (by rotation around the left-most extremity of the line). That's why the position of the instructions `\Ddots` in the array can have a marked effect on the final result.

In the following examples, the first `\Ddots` instruction is written in color.

Example with parallelization (default):

```
$A = \begin{pNiceMatrix}
  1      & \Cdots &        & 1      \\
  a+b    & \Ddots &        & \Vdots \\
  \Vdots & \Ddots &        &        \\
  a+b    & \Cdots & a+b    & 1
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} 1 & \cdots & & 1 \\ a+b & \ddots & & \vdots \\ \vdots & & \ddots & \\ a+b & \cdots & a+b & 1 \end{pmatrix}$$

```
$A = \begin{pNiceMatrix}
  1      & \Cdots &        & 1      \\
  a+b    &        &        & \Vdots \\
  \Vdots & \Ddots & \Ddots &        \\
  a+b    & \Cdots & a+b    & 1
\end{pNiceMatrix}$
```

$$A = \begin{pmatrix} 1 & \cdots & & 1 \\ a+b & \ddots & & \vdots \\ \vdots & & \ddots & \\ a+b & \cdots & a+b & 1 \end{pmatrix}$$

It's possible to turn off the parallelization with the option `parallelize-diags` set to `false`:

The same example without parallelization:

$$A = \begin{pmatrix} 1 & \cdots & & 1 \\ a+b & \ddots & & \vdots \\ \vdots & & \ddots & \\ a+b & \cdots & a+b & 1 \end{pmatrix}$$

It's possible to specify the instruction `\Ddots` which will be drawn first (and which will be used to draw the other diagonal dotted lines when the parallelization is in force) with the key `draw-first`: `\Ddots[draw-first]`.

[75] We speak of the lines created by `\Ddots` or `\Iddots` and not the lines created by a command `\line` in the `\CodeAfter`.

17.2 The "empty" cells

The package nicematrix uses, in several circunstancies, the concept of "empty cell". For example, an instruction like `\Ldots`, `\Cdots`, etc. (cf. 10, p. 38) tries to determine the first non-empty cell on both sides. In the same way, when the key `corners` is used (cf. 5.3.3, p. 17), nicematrix computes corners consisting of empty cells.
However, an "empty cell" is not necessarily a cell with no TeX content (that is to say a cell with no token between the two ampersands `&`). The precise rules are as follow.

- An implicit cell is empty. For example, in the following matrix:

  ```
  \begin{pmatrix}
  a & b \\
  c \\
  \end{pmatrix}
  ```

  the last cell (second row and second column) is empty.

- For the columns of type `p`, `m`, `b`, `V`[76] and `X`[77], the cell is empty if (and only if) its content in the TeX code is empty (there is only spaces between the ampersands `&`).

- For the columns of type `c`, `l`, `r` and `w{...}{...}`, the cell is empty if (and only if) its TeX output has a width equal to zero.

- A cell containing the command `\NotEmpty` is not empty (and a PGF/TikZ node is created in that cell).

- A cell within a block (command `\Block`) which uses the key `empty` is considered as empty.

- A cell with only a command `\Hspace` (or `\Hspace*`) is empty. This command `\Hspace` is a command defined by the package nicematrix with the same meaning as `\hspace` except that the cell where it is used is considered as empty. This command can be used to fix the width of some columns of the matrix without interfering with nicematrix.

17.3 The option exterior-arraycolsep

The environment `{array}` inserts an horizontal space equal to `\arraycolsep` before and after each column. In particular, there is a space equal to `\arraycolsep` before and after the array. This feature of the environment `{array}` was probably not a good idea[78]. The environment `{matrix}` of amsmath and its variants (`{pmatrix}`, `{vmatrix}`, etc.) of amsmath prefer to delete these spaces with explicit instructions `\hskip -\arraycolsep`[79]. The package nicematrix does the same in all its environments, `{NiceArray}` included. However, if the user wants the environment `{NiceArray}` behaving by default like the environment `{array}` of array (for example, when adapting an existing document) it's possible to control this behaviour with the option `exterior-arraycolsep`, set by the command `\NiceMatrixOptions`. With this option, exterior spaces of length `\arraycolsep` will be inserted in the environments `{NiceArray}` (the other environments of nicematrix are not affected).

[76] The columns of type `V` are provided by varwidth, which must be loaded: cf. 8.2, p. 34.

[77] See 8.3, p. 35

[78] In the documentation of amsmath, we can read: *The extra space of* `\arraycolsep` *that array adds on each side is a waste so we remove it [in* `{matrix}`*] (perhaps we should instead remove it from array in general, but that's a harder task).*

[79] And not by inserting `@{}` on both sides of the preamble of the array. As a consequence, the length of the `\hline` is not modified and may appear too long, in particular when using square brackets.

17.4 Incompatibilities

The package nicematrix can't be used in the class revtex4-2 because this class is not compatible with the recent versions of LaTeX. Indeed, revtex4-2 modifies serval interal commands of array and those modifications have not been updated to take into acount the recent changes in array in relation with the Tagging Project.

There might be incompatibilities between nicematrix and babel with the language which activate (in the TeX sens) some characters, in particular the character <.
Par exemple, for Spanish, it's recommended to deactivate the abreviations at load-time with the instruction:

```
\usepackage[spanish,es-noshorthands]{babel}
```

The extension nicematrix is usually not compatible with the classes and packages that redefine the environments `{tabular}` and `{array}`. In particular, it's the case of the class `socg-lipics-v2021`. However, in that case, it's possible to load the class with the key `notab` which requires that the environment `{tabular}` is not redefined.

The package nicematrix is not compatible with the class ieeeaccess because that class is not compatible with PGF/TikZ. However, there is a simple workaround by writing:[80]

```
\let\TeXyear\year
\documentclass{IEEEaccess}
\let\year\TeXyear
```

In order to use nicematrix with the class aastex631 (of the *American Astronomical Society*), you have to add the following instructions in the preamble of your document :

```
\BeforeBegin{NiceTabular}{\let\begin\BeginEnvironment\let\end\EndEnvironment}
\BeforeBegin{NiceArray}{\let\begin\BeginEnvironment}
\BeforeBegin{NiceMatrix}{\let\begin\BeginEnvironment}
```

The package nicematrix is not fully compatible with the packages and classes of LuaTeX-ja: the detection of the empty corners (cf. 5.3.3, p. 17) may be wrong in some circonstances.

The package nicematrix is not compatible with the package arydshln and does not support the columns `V` of boldline (because the letter `V` is reserved for the columns `V` of varwidth). By any means, nicematrix provides tools to draw dashed rules (with the command `\hdashedline` and the letter `;`) and rules of different weights with the key `custom-line` (cf. 5.3.5, p. 19).

The columns `d` of dcolumn and the columns `n` of numprint are not supported (but it's possible to use the columns `S` of siunitx).

17.5 Compatibility with the Tagging Project of LaTeX

Since the version 7.0, the extension nicematrix is compatible with the *Tagging Project* of LaTeX (whose aim is to create tagged PDF). At this time, only simple standard tabulars are correctly tagged.
Here is a example of code which will generate a tagged PDF.

```
\DocumentMetadata{tagging = on}
\documentclass{article}
\usepackage{lmodern}
\usepackage{nicematrix}

\begin{document}

\begin{center}
\tagpdfsetup{table/header-rows=1}
\begin{NiceTabular}{ccc}[hvlines]
  First name & Last name & Age \\
```

[80] See `https://tex.stackexchange.com/questions/528975`

```
  Paul       & Imbert    & $66$ \\
  John       & Sarrus     & $23$ \\
  Liz        & Taylor     & $100$ \\
  George     & Adams      & $34$
\end{NiceTabular}
\end{center}

\end{document}
```



18 Examples

18.1 Notes in the tabulars

The tools provided by nicematrix for the composition of the tabular notes have been presented in the section 13, p. 53.

Let's consider that we wish to number the notes of a tabular with stars.[81]

First, we write a command `\stars` similar to the well-known commands `\arabic`, `\alph`, `\Alph`, etc. which produces a number of stars equal to its argument[82].

```
\ExplSyntaxOn
\NewDocumentCommand { \stars } { m }
  { \prg_replicate:nn { \value { #1 } } { \( \star \) } }
\ExplSyntaxOff
```



Of course, we change the style of the labels with the key `notes/style`. However, it would be interesting to change also some parameters in the type of list used to compose the notes at the end of the tabular. First, we required a composition flush right for the labels with the setting `align=right`. Moreover, we want the labels to be composed on a width equal to the width of the widest label. The widest label is, of course, the label with the greatest number of stars. We know that number: it is equal to `\value{tabularnote}` (because `tabularnote` is the LaTeX counter used by `\tabularnote` and, therefore, at the end of the tabular, its value is equal to the total number of tabular notes). We use the key `widest*` of enumitem in order to require a width equal to that value: `widest*=\value{tabularnote}`.

```
\NiceMatrixOptions
  {
    notes =
     {
       style = \stars{#1} ,
       enumitem-keys =
        {
          widest* = \value{tabularnote} ,
          align = right
        }
     }
  }
\begin{NiceTabular}{@{}llr@{}}
\toprule \RowStyle{\bfseries}
  Last name & First name & Birth day \\
\midrule
  Achard\tabularnote{Achard is an old family of the Poitou.}
  & Jacques & 5 juin 1962 \\
  Lefebvre\tabularnote{The name Lefebvre is an alteration of the name Lefebure.}
```



[81] Of course, it's realistic only when there is very few notes in the tabular.
[82] In fact: the value of its argument.

```
  & Mathilde & 23 mai 1988 \\
  Vanesse & Stephany & 30 octobre 1994 \\
  Dupont & Chantal & 15 janvier 1998 \\
\bottomrule
\end{NiceTabular}
```

Last name	First name	Birth day
Achard*	Jacques	5 juin 1962
Lefebvre**	Mathilde	23 mai 1988
Vanesse	Stephany	30 octobre 1994
Dupont	Chantal	15 janvier 1998

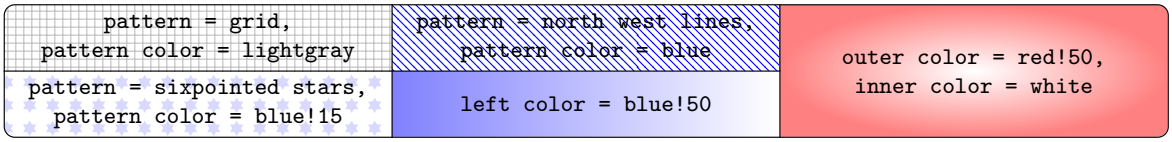
*Achard is an old family of the Poitou.
**The name Lefebvre is an alteration of the name Lefebure.

18.2 Use of the key "tikz" of the command \Block

The key `tikz` of the command `\Block` is available only when TikZ is loaded.[83]
For the following example, we also need the TikZ library `patterns`.

```
\usepackage{tikz}        % in the preamble
\usetikzlibrary{patterns} % in the preamble

\ttfamily \small
\begin{NiceTabular}{X[m]X[m]X[m]}[hvlines,cell-space-limits=3pt,rounded-corners]
  \Block[tikz={pattern=grid,pattern color=lightgray}]{}
    {pattern = grid,\\ pattern color = lightgray}
& \Block[tikz={pattern = north west lines,pattern color=blue}]{}
    {pattern = north west lines,\\ pattern color = blue}
& \Block[tikz={outer color = red!50, inner color=white }]{2-1}
    {outer color = red!50,\\ inner color = white} \\
  \Block[tikz={pattern = sixpointed stars, pattern color = blue!15}]{}
    {pattern = sixpointed stars,\\ pattern color = blue!15}
& \Block[tikz={left color = blue!50}]{}
    {left color = blue!50} \\
\end{NiceTabular}
```



In the following example, we use the key `tikz` to hatch a row of the tabular. Remark that you use the key `transparent` of the command `\Block` in order to have the rules drawn in the block.[84]

```
\begin{NiceTabular}{ccc}[hvlines]
\CodeBefore
  \columncolor[RGB]{169,208,142}{2}
\Body
  one & two & three \\
  \Block[transparent,tikz={pattern = north west lines, pattern color = red}]{1-*}{}
  four & five & six \\
  seven & eight & nine
\end{NiceTabular}
```

[83] By default, nicematrix only loads PGF, which is a sub-layer of TikZ.
[84] By default, the rules are not drawn in the blocks created by the command `\Block` (cf. 5, p. 14): with the key `transparent`, they are drawn (the block becomes *transparent* to the exterior rules).

one	two	three
four	five	six
seven	eight	nine

18.3 Use with tcolorbox

Here is an example of use of `{NiceTabular}` within a command `\tcbox` of tcolorbox. We have used the key `hvlines-except-borders` in order to draw all the rules, except on the borders (which are, of course, added by tcolorbox).

```
\tcbset
  {
    colframe = blue!50!black ,
    colback = white ,
    fonttitle = \bfseries ,
    nobeforeafter ,
    center title
  }
\tcbox
  [
    left = 0mm ,
    right = 0mm ,
    top = 0mm ,
    bottom = 0mm ,
    boxsep = 0mm ,
    toptitle = 0.5mm ,
    bottomtitle = 0.5mm ,
    title = My table
  ]
  {
    \renewcommand{\arraystretch}{1.2}% <-- the % is mandatory here
    \begin{NiceTabular}{rcl}[hvlines-except-borders,rules/color=blue!50!black]
    \CodeBefore
      \rowcolor{red!15}{1}
    \Body
      One & Two & Three \\
      Men & Mice & Lions \\
      Upper & Middle & Lower
    \end{NiceTabular}
  }
```



My table

One	Two	Three
Men	Mice	Lions
Upper	Middle	Lower

18.4 A sign chart

```
\color{blue}
\[\begin{NiceArray}
  [columns-width=1cm,hvlines,rules/color=black]
  {>{\color{black}}cccccc}
  f(x)   & \Block[C]{1-5}{+} \\
  f'(x)  & \Block[C]{1-2}{+} && \Block[C]{1-2}{+} && + \\
  f''(x) & - & \Block[C]{1-2}{-} && \Block[C]{1-2}{+} \\
\end{NiceArray}\]
```



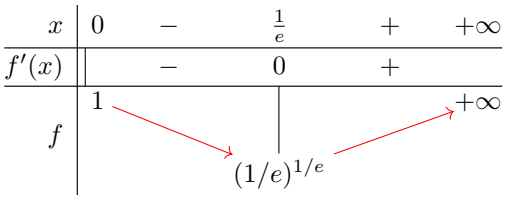
$f(x)$			+				
$f'(x)$		+			+		+
$f''(x)$	−			−		+	

With the key `C` of the command `\Block`, the content of the block is centered according to the absolute limits of the columns (whereas, with the key `c`, the content is centered according to the *contents* of the cells of the corresponding columns — and, here, there are many columns without content).

18.5 A variation table

In the following variation table, you draw the arrows (and some additional rules) with TikZ by using the PGF/TikZ nodes created by nicematrix (cf. 15, p. 64).

```
$\begin{NiceArray}{r|cw{c}{1cm}cw{c}{1cm}c}[cell-space-limits=2pt]
x     & 0 & - & \frac{1}{e} & +  & +\infty \\
\Hline
f'(x) &   & - & 0 & + & \\
\Hline
      & 1 &   &   &   & +\infty \\
f     &   &   &   &   &         \\
      &   &   & (1/e)^{1/e} \\
\CodeAfter
  \begin{tikzpicture}
  \draw (3-|4.5) -- (5-|4.5) ;
  \draw ([xshift=3pt]2-|2) -- ([xshift=3pt]3-|2) ;
  \draw [->,red,shorten < = 3pt] (3-2) -- (5-4) ;
  \draw [->,red,shorten < = 3pt] (5-4) -- (3-6) ;
  \end{tikzpicture}
\end{NiceArray}$
```



18.6 Example of description of a table of a database

In the following example, we use the extension hhline (it would have been also possible to use the package ehhline) in an environment `{NiceTabular}` in order to draw the double horizontal rule which is conventionnaly used to denote the primary key of the table.

```
\usepackage{hhline} % in the preamble

\begin{NiceTabular}{>{\columncolor{red!15}}cllr}[vlines]
\Hline
key  & first name & last name & age \\
\hhline{=---}
2072 & Jean       & Marcier   & 68  \\
2073 & Claude     & Beaumont  & 23  \\
\Hline
\end{NiceTabular}
```

key	first name	last name	age
2072	Jean	Marcier	68
2073	Claude	Beaumont	23

As we see, thanks to the use of the key `vlines`, we have been able to write `\hhline{=---}` instead of `\hhline{|=|-|-|-|}`.

18.7 Use of the key borders of the command \Block

In the following example, we use the command \Block to draw rules.

```
\usepackage{tikz} % in the preamble

$\begin{pNiceMatrix}
  \Block[borders={bottom,right,tikz=dotted}]{2-2}{}
  1 & 2 & 0  & 0 & 0 & 0 \\
  4 & 5 & 0  & 0 & 0 & 0 \\
  0 & 0 & \Block[borders={all,tikz=dotted}]{2-2}{}
          7  & 1 & 0 & 0 \\
  0 & 0 & -1 & 2 & 0 & 0 \\
  0 & 0 & 0  & 0 & \Block[borders={left,top,tikz=dotted}]{2-2}{}
                   3 & 4 \\
  0 & 0 & 0  & 0 & 1 & 4
\end{pNiceMatrix}$
```

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 1 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 0 & 1 & 4 \end{pmatrix}$$

18.8 Leaders in the matrices

An example with the resultant of two polynoms (this example requires TikZ for the dashed rule specified by the letter "`;`" in the preamble of the array).

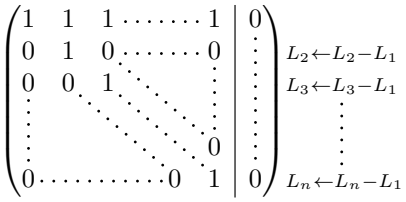
```
\usepackage{tikz}      % in the preamble

\setlength{\extrarowheight}{1mm}
\[\begin{vNiceArray}{cccc;ccc}[columns-width=6mm, xdots/shorten += 3pt]
  a_0   &      &&       &b_0    &      &      \\
  a_1   &\Ddots&&       &b_1    &\Ddots&      \\
  \Vdots&\Ddots&&       &\Vdots &\Ddots&b_0   \\
  a_p   &      &&a_0    &       &      &b_1   \\
        &\Ddots&&a_1    &b_q    &      &\Vdots\\
        &      &&\Vdots &       &\Ddots&      \\
        &      &&a_p    &       &      &b_q
\end{vNiceArray}\]
```

$$\left|\begin{array}{ccc:ccc} a_0 & & & b_0 & & \\ a_1 & \ddots & & b_1 & \ddots & \\ \vdots & \ddots & a_0 & \vdots & \ddots & b_0 \\ a_p & \ddots & a_1 & b_q & \ddots & b_1 \\ & \ddots & \vdots & & \ddots & \vdots \\ & & a_p & & & b_q \end{array}\right|$$

An example for a linear system. We recall that the key **xdots/nullify** nullifies the horizontal and vertical spaces created by the commands **\Cdots**, **\Vdots**, etc. in the tabular (but the dotted lines are actually drawn, despite the name **nullify**!).

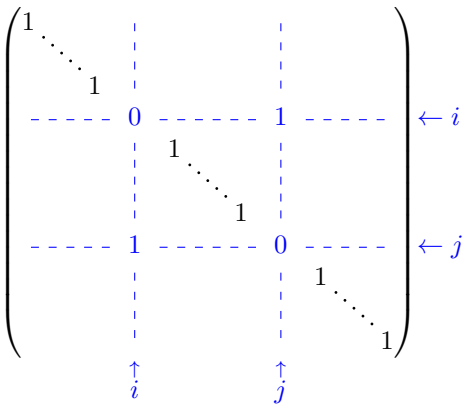
```
\NiceMatrixOptions{code-for-last-col=\scriptstyle}
$\begin{pNiceMatrix}[vlines=-1,xdots/nullify,last-col]
  1      & 1 & 1 &\Cdots &   & 1      & 0      & \\
  0      & 1 & 0 &\Cdots &   & 0      &        & L_2 \gets L_2-L_1 \\
  0      & 0 & 1 &\Ddots &   & \Vdots &        & L_3 \gets L_3-L_1 \\
         &   &   &\Ddots &   &        & \Vdots & \Vdots \\
  \Vdots &   &   &\Ddots &   & 0      & \\
  0      &   &   &\Cdots & 0 & 1      & 0      & L_n \gets L_n-L_1
\end{pNiceMatrix}$
```



The option **line-style** controls the style of the leaders drawn by **\Ldots**, **\Cdots**, etc. Thus, it's possible with these commands to draw lines which are not longer dotted.

```
\usepackage{tikz}             % in the preamble
\usetikzlibrary{decorations}  % in the preamble


\setcounter{MaxMatrixCols}{12}
\newcommand{\blue}{\color{blue}}
\[\begin{pNiceMatrix}%
    [last-row,last-col,xdots={nullify,line-style={dashed,dash expand off,blue}}]
  1& & & \Vdots & & & & \Vdots \\
  & \Ddots[line-style=standard] \\
  & & 1 \\
  \Cdots & & & \blue 0 & \Cdots & & & \blue 1 & & & \Cdots & \blue \leftarrow i \\
  & & & & 1 \\
  & & &\Vdots & & \Ddots[line-style=standard] & & \Vdots \\
  & & & & & & 1 \\
  \Cdots & & & \blue 1 & \Cdots & & \Cdots & \blue 0 & & & \Cdots & \blue \leftarrow j \\
  & & & & & & & & 1 \\
  & & & & & & & & & \Ddots[line-style=standard] \\
  & & & \Vdots & & & & \Vdots & & & 1 \\
  & & & \blue \overset{\uparrow}{i} & & & & \blue \overset{\uparrow}{j} \\
\end{pNiceMatrix}\]
```



18.9 How the illustrate the numbers of rows and columns in a matrix

In this document, the TikZ library arrows.meta has been loaded, which impacts the shape of the arrow tips.

```
\NiceMatrixOptions{xdots={horizontal-labels,line-style = <->}}
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hdotsfor{3}^{3} & \Hdotsfor{2}^{2} \\
  2 & 1 & 1 & 1 & 1 & \Vdotsfor{3}^{3}\\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
\Hline
  1 & 1 & 1 & 1 & 1 & \Vdotsfor{2}^{2} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```

$$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hdotsfor{3}^{3} & \Hdotsfor{2}^{2} \\
  2 & 1 & 1 & 1 & 1 & \Vdotsfor{3}^{3}\\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
\Hline
  1 & 1 & 1 & 1 & 1 & \Vdotsfor{2}^{2} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$$

If you want the label *on the line*, you should use the special token "`:`" instead of "`^`":

```
\NiceMatrixOptions{xdots={horizontal-labels,line-style = <->}}
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hdotsfor{3}:{3} & \Hdotsfor{2}:{2} \\
  2 & 1 & 1 & 1 & 1 & \Vdotsfor{3}:{3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
\Hline
  1 & 1 & 1 & 1 & 1 &
  \Vdotsfor{2}:{2 \rlap{ \smash{rows}}} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```

$$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hdotsfor{3}:{3} & \Hdotsfor{2}:{2} \\
  2 & 1 & 1 & 1 & 1 & \Vdotsfor{3}:{3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
\Hline
  1 & 1 & 1 & 1 & 1 &
  \Vdotsfor{2}:{2 \rlap{ \smash{rows}}} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$$

If one prefers the braces of the library decorations.pathreplacing of TikZ, the best way is to use the commands `\Hbrace` and `\Vbrace` provided by nicematrix (cf. 10.7, p. 44).

```
\usepackage{tikz}                          % in the preamble
\usetikzlibrary{decorations.pathreplacing} % in the preamble
```

```
\NiceMatrixOptions{xdots/horizontal-labels}
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hbrace{3}{3} & \Hbrace{2}{2} \\
  2 & 1 & 1 & 1 & 1 & \Vbrace{3}{3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
\Hline
  1 & 1 & 1 & 1 & 1 & \Vbrace{2}{2} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```

$$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hbrace{3}{3} & \Hbrace{2}{2} \\
  2 & 1 & 1 & 1 & 1 & \Vbrace{3}{3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
\Hline
  1 & 1 & 1 & 1 & 1 & \Vbrace{2}{2} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$$

It's possible to change the TikZ style of those braces by changing the TikZ style `nicematrix/brace`. In the following example, we use "calligraphic" braces (the TikZ library calligraphy must be loaded *after* decorations.pathreplacing).

```
\usepackage{tikz}                                    % in the preamble
\usetikzlibrary{decorations.pathreplacing,calligraphy} % in the preamble
```

```
\NiceMatrixOptions{xdots/horizontal-labels}
\tikzset
  {
    nicematrix/brace/.style =
      {
        decoration =
          {
            calligraphic brace ,
            amplitude = 0.4 em ,
            raise = -0.25 em
          } ,
        line width = 0.1 em ,
        decorate
      }
  }
$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
  \Hbrace{3}{3} & \Hbrace{2}{2} \\
  2 & 1 & 1 & 1 & 1 & \Vbrace{3}{3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
\Hline
  1 & 1 & 1 & 1 & 1 & \Vbrace{2}{2} \\
  1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$
```

$$\begin{pNiceArray}{ccc|cc}[first-row,last-col,margin]
\Hbrace{3}{3} & \Hbrace{2}{2} \\
2 & 1 & 1 & 1 & 1 & \Vbrace{3}{3} \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\Hline
1 & 1 & 1 & 1 & 1 & \Vbrace{2}{2} \\
1 & 1 & 1 & 1 & 1 \\
\end{pNiceArray}$$

If one prefers the curly braces of the current mathematical font of LaTeX, one should use the commands `\SubMatrix`, `\OverBrace` and `\UnderBrace` in the `\CodeAfter`.

```
$\begin{pNiceArray}{ccc|cc}[margin,last-col]
  2 & 1 & 1 & 1 & 1 & \Block{3-1}{\quad 3} \\
  1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 \\
\Hline
  1 & 1 & 1 & 1 & 1 & \Block{2-1}{\quad 2} \\
  1 & 1 & 1 & 1 & 1 \\
\CodeAfter
  \OverBrace[shorten,yshift=1.5mm]{1-1}{1-3}{3}
  \OverBrace[shorten,yshift=1.5mm]{1-4}{1-5}{2}
  \SubMatrix{.}{1-1}{3-5}{\rbrace}[xshift=3.5mm]
  \SubMatrix{.}{4-1}{5-5}{\rbrace}[xshift=3.5mm]
\end{pNiceArray}$
```

$$\begin{pNiceArray}{ccc|cc}[margin,last-col]
2 & 1 & 1 & 1 & 1 & \Block{3-1}{\quad 3} \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\Hline
1 & 1 & 1 & 1 & 1 & \Block{2-1}{\quad 2} \\
1 & 1 & 1 & 1 & 1 \\
\CodeAfter
\OverBrace[shorten,yshift=1.5mm]{1-1}{1-3}{3}
\OverBrace[shorten,yshift=1.5mm]{1-4}{1-5}{2}
\SubMatrix{.}{1-1}{3-5}{\rbrace}[xshift=3.5mm]
\SubMatrix{.}{4-1}{5-5}{\rbrace}[xshift=3.5mm]
\end{pNiceArray}$$

Or course, the output may seem disappointing. That's why, for this type of use, we recommend the use of the commands `\Hbrace` and `\Vbrace` (provided by nicematrix), as shown in the previous examples.

18.10 Stacks of matrices

We often need to compose mathematical matrices on top on each other (for example for the resolution of linear systems).

In order to have the columns aligned one above the other, it's possible to fix a width for all the columns. That's what is done in the following example with the environment `{NiceMatrixBlock}` and its option `auto-columns-width`.

```
\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
  {
    light-syntax,
```

```
    last-col, code-for-last-col = \color{blue}\scriptstyle,
    vlines = 5 ,
    matrix/columns-type = r ,
    no-cell-nodes % only for speedup
  }
\setlength{\extrarowheight}{1mm}

\quad $\begin{pNiceMatrix}
12  -8  7  5   3 {} ;
 3 -18 12  1   4    ;
-3 -46 29 -2 -15    ;
 9  10 -5  4   7
\end{pNiceMatrix}$

\smallskip
\quad $\begin{pNiceMatrix}
12  -8   7  5   3 ;
0   64 -41  1  19  { L_2 \gets L_1-4L_2  } ;
0 -192 123 -3 -57  { L_3 \gets L_1+4L_3  } ;
0 -64   41 -1 -19  { L_4 \gets 3L_1-4L_4 } ;
\end{pNiceMatrix}$

\smallskip
\quad $\begin{pNiceMatrix}
12 -8   7 5  3 ;
0  64 -41 1 19 ;
0   0   0 0  0 { L_3 \gets 3 L_2 + L_3 }
\end{pNiceMatrix}$

\smallskip
\quad $\begin{pNiceMatrix}
12 -8   7 5  3 {} ;
0  64 -41 1 19    ;
\end{pNiceMatrix}$
\end{NiceMatrixBlock}
```



$$\left(\begin{array}{rrrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

$$\left(\begin{array}{rrrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right) \begin{array}{l} \\ L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}$$

$$\left(\begin{array}{rrrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) L_3 \leftarrow 3L_2 + L_3$$

$$\left(\begin{array}{rrrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

However, as can be seen, the last matrix is not perfectly aligned with others. That's because, in LaTeX, the parenthesis have not exactly the same width (smaller parenthesis are a bit slimer).

In order the solve that problem, it's possible to require the delimiters to be composed with the maximal width, thanks to the boolean key `delimiters/max-width`.

```
\begin{NiceMatrixBlock}[auto-columns-width]
\NiceMatrixOptions
  {
    delimiters/max-width,
```

```
    light-syntax,
    last-col, code-for-last-col = \color{blue}\scriptstyle,
    vlines = 5 ,
    matrix/columns-type = r ,
    no-cell-nodes % only for speedup
  }
\setlength{\extrarowheight}{1mm}

\quad $\begin{pNiceMatrix}
12  -8  7  5   3 {} ;
 3 -18 12  1   4    ;
-3 -46 29 -2 -15    ;
 9  10 -5  4   7
\end{pNiceMatrix}$
etc.
\end{NiceMatrixBlock}
```



$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right) \begin{array}{l} \\ L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) L_3 \leftarrow 3L_2 + L_3$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

If you wish an alignment of the different matrices without the same width for all the columns, you can construct a unique array and place the parenthesis with commands `\SubMatrix` in the `\CodeAfter`. Of course, that array can't be broken by a page break.

```
\setlength{\extrarowheight}{1mm}
\[\begin{NiceMatrix}[r, last-col=6, code-for-last-col = \scriptstyle \color{blue}]
12 &  -8  &  7 & 5 &  3 \\
 3 & -18  & 12 & 1 &  4 \\
-3 & -46  & 29 &-2 &-15 \\
 9 & 10   &-5  &4  & 7 \\[1mm]
12 & -8   & 7  &5  & 3 \\
0  & 64   &-41 & 1 & 19 & L_2 \gets L_1-4L_2  \\
0  & -192 &123 &-3 &-57 & L_3 \gets L_1+4L_3  \\
0  & -64  & 41 &-1 &-19 & L_4 \gets 3L_1-4L_4 \\[1mm]
12 & -8   &7   &5  & 3 \\
0  & 64   &-41 &1  &19 \\
0  &  0   &0   &0  & 0  & L_3 \gets 3L_2+L_3 \\[1mm]
12 & -8   &7   &5  & 3 \\
0  & 64   &-41 & 1 & 19 \\
\CodeAfter [sub-matrix/vlines=4]
   \SubMatrix({1-1}{4-5})
   \SubMatrix({5-1}{8-5})
   \SubMatrix({9-1}{11-5})
   \SubMatrix({12-1}{13-5})
\end{NiceMatrix}\]
```



$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right) \begin{array}{l} \\ L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \begin{array}{l} \\ \\ \\ L_3 \leftarrow 3L_2 + L_3 \end{array}$$

$$\left(\begin{array}{rrrr|r} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

In this tabular, the instructions `\SubMatrix` are executed after the composition of the tabular and, thus, the vertical rules are drawn without adding space between the columns.

In fact, it's possible, with the key `vlines-in-sub-matrix`, to choice a letter in the preamble of the array to specify vertical rules which will be drawn in the `\SubMatrix` only (and it will add space between the columns for that rule).

```
\setlength{\extrarowheight}{1mm}
\[\begin{NiceArray}
   [
     vlines-in-sub-matrix = I,
     last-col,
     code-for-last-col = \scriptstyle \color{blue}
   ]
  {rrrrIr}
12 &  -8  &  7 & 5 &  3 \\
 3 & -18  & 12 & 1 &  4 \\
-3 & -46  & 29 &-2 &-15 \\
 9 & 10   &-5  &4  & 7 \\[1mm]
12 & -8   & 7  &5  & 3 \\
0  & 64   &-41 & 1 & 19 & L_2 \gets L_1-4L_2  \\
0  & -192 &123 &-3 &-57 & L_3 \gets L_1+4L_3  \\
0  & -64  & 41 &-1 &-19 & L_4 \gets 3L_1-4L_4 \\[1mm]
12 & -8   &7   &5  & 3 \\
0  & 64   &-41 &1  &19 \\
0  &  0   &0   &0  & 0 & L_3 \gets 3L_2+L_3 \\[1mm]
12 & -8   &7   &5  & 3 \\
0  & 64   &-41 & 1 & 19 \\
\CodeAfter
   \SubMatrix({1-1}{4-5})
   \SubMatrix({5-1}{8-5})
   \SubMatrix({9-1}{11-5})
   \SubMatrix({12-1}{13-5})
\end{NiceArray}\]
```



$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 3 & -18 & 12 & 1 & 4 \\ -3 & -46 & 29 & -2 & -15 \\ 9 & 10 & -5 & 4 & 7 \end{array}\right)$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & -192 & 123 & -3 & -57 \\ 0 & -64 & 41 & -1 & -19 \end{array}\right) \begin{array}{l} \\ L_2 \leftarrow L_1 - 4L_2 \\ L_3 \leftarrow L_1 + 4L_3 \\ L_4 \leftarrow 3L_1 - 4L_4 \end{array}$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \begin{array}{l} \\ \\ \\ L_3 \leftarrow 3L_2 + L_3 \end{array}$$

$$\left(\begin{array}{cccc|c} 12 & -8 & 7 & 5 & 3 \\ 0 & 64 & -41 & 1 & 19 \end{array}\right)$$

18.11 How to highlight cells of a matrix

In order to highlight a cell of a matrix, it's possible to "draw" that cell with the key `draw` of the command `\Block` (this is one of the uses of a mono-cell block[85]).

```
$\begin{pNiceArray}{>{\strut}cccc}[margin,rules/color=blue,no-cell-nodes]
\Block[draw]{}{a_{11}} & a_{12} & a_{13} & a_{14} \\
a_{21} & \Block[draw]{}{a_{22}} & a_{23} & a_{24} \\
a_{31} & a_{32} & \Block[draw]{}{a_{33}} & a_{34} \\
a_{41} & a_{42} & a_{43} & \Block[draw]{}{a_{44}} \\
\end{pNiceArray}$
```

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

We should remark that the rules we have drawn are drawn *after* the construction of the array and thus, they don't spread the cells of the array. We recall that, on the other side, the commands `\hline` and `\Hline`, the specifier "`|`" and the options `hlines`, `vlines`, `hvlines`, `hvlines-except-borders` and `hlines-except-borders` spread the cells.[86]

It's possible to color a row with `\rowcolor` in the `\CodeBefore`, or with `\rowcolor` in the first cell of the row (if used in another cell of the row `\rowcolor` will color till the end of the row).

```
$\begin{pNiceArray}{>{\strut}cccc}% <-- % mandatory
   [margin, extra-margin=2pt,no-cell-nodes]
  \rowcolor{red!15}A_{11} & A_{12} & A_{13} & A_{14} \\
  A_{21} & \rowcolor{red!15}A_{22} & A_{23} & A_{24} \\
  A_{31} & A_{32} & \rowcolor{red!15}A_{33} & A_{34} \\
  A_{41} & A_{42} & A_{43} & \rowcolor{red!15}A_{44}
\end{pNiceArray}$
```

[85] We recall that, if the first mandatory argument of the command `\Block` is left empty, that means that the block is a mono-cell block.

[86] For the command `\cline`, see the remark in the section 5.1.2, p. 15.

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{pmatrix}$$

For a more precise tuning, it's possible to use the command `\Block`.

```
\usepackage{tikz} % in the preamble

\NewDocumentCommand{\OneRow}{}
  {\Block[tikz={xoffset=2.5pt,yoffset=0.8pt,fill=red!15,rounded corners = 2pt}]{1-*}{}}
$\begin{pNiceArray}{ccc}[last-col, margin = 2pt]
\OneRow a & a + b & a + b + c & L_1 \\
\OneRow a & a     & a + b     & L_2 \\
\OneRow a & a     & a         & L_3
\end{pNiceArray}$
```

$$\begin{pmatrix} a & a+b & a+b+c \\ a & a & a+b \\ a & a & a \end{pmatrix} \begin{matrix} L_1 \\ L_2 \\ L_3 \end{matrix}$$

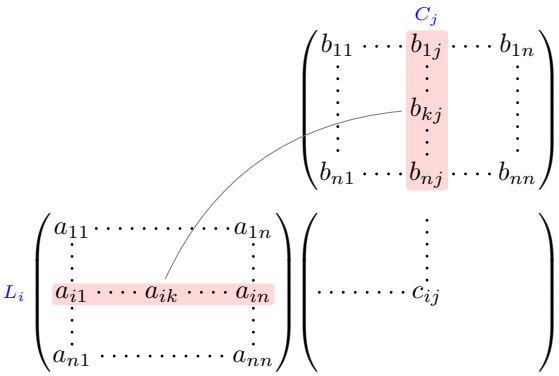
However, that technic is sometimes not sufficient when we have to underline a part of a row or a part of a column.
That's why we present now a more general technic.
First, we load the TikZ library fit and we define a TikZ style called `highlight`.

```
\usepackage{tikz} % dans le préambule
\usetikzlibrary{fit} % dans le préambule

\tikzset{highlight/.style={rectangle,
                           fill=red!15,
                           rounded corners = 0.5 mm,
                           inner sep=1pt,
                           fit = #1}}
```

We use that style in the `\CodeBefore` of the environment.

```
\[\begin{NiceArray}{*{6}{c}@{\hspace{6mm}}*{5}{c}}[xdots/nullify]
\CodeBefore [create-cell-nodes]
  \SubMatrix({2-7}{6-last})
  \SubMatrix({7-2}{last-6})
  \SubMatrix({7-7}{last-last})
  \begin{tikzpicture}
    \node [highlight = (9-2) (9-6)] { } ;
    \node [highlight = (2-9) (6-9)] { } ;
  \end{tikzpicture}
\Body
    &        &        &        &        &        &        & \color{blue}\scriptstyle C_j \\
    &        &        &        &        &        & b_{11} & \Cdots & b_{1j} & \Cdots & b_{1n} \\
    &        &        &        &        &        & \Vdots &        & \Vdots &        & \Vdots \\
    &        &        &        &        &        &        &        & b_{kj} \\
    &        &        &        &        &        &        &        & \Vdots \\
    &        &        &        &        &        & b_{n1} & \Cdots & b_{nj} & \Cdots & b_{nn} \\[3mm]
    & a_{11} & \Cdots &        &        & a_{1n} \\
    & \Vdots &        &        &        & \Vdots &        &        & \Vdots \\
\color{blue}\scriptstyle L_i
    & a_{i1} & \Cdots & a_{ik} & \Cdots & a_{in} & \Cdots &        & c_{ij} \\
    & \Vdots &        &        &        & \Vdots \\
    & a_{n1} & \Cdots &        &        & a_{nn}  \\
\CodeAfter
\tikz \draw [gray,shorten > = 1mm, shorten < = 1mm] (9-4.north) to [bend left] (4-9.west) ;
\end{NiceArray}\]
```



The whole figure is an environment `{NiceArray}` and the three pairs of parenthesis have been added with `\SubMatrix` in the `\CodeBefore` (since you have put these `\SubMatrix` in the `\CodeBefore` — and not the `\CodeAfter` — the dotted leaders specified in the matrix product C will be limited to that matrix).

18.12 A triangular tabular

In the following example, we use the style PGF/TikZ `nicematrix/cell-node` (cf. 15.1) to rotate the contents of the cells (and, then, we compensate that rotation by a rotation of the whole tabular with the command `\adjustbox` of the eponymous package, which must be loaded previously).

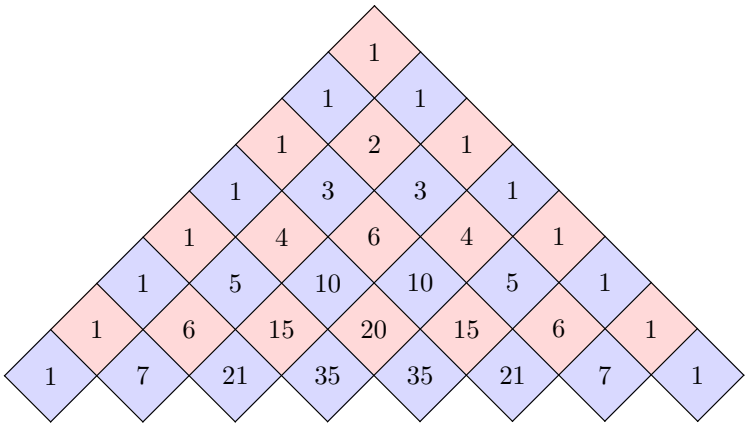
```
\usepackage{adjustbox} % in the preamble


\pgfset
  {
    nicematrix/cell-node/.append style =
      { text/rotate = 45, minimum size = 6 mm }
  }

\setlength{\tabcolsep}{0pt}

\adjustbox{rotate = -45, set depth = 6mm + 1.414 \arrayrulewidth}
  {\begin{NiceTabular} [ hvlines, corners=S, baseline = line-9 ] { cccccccc }
   \CodeBefore
     \chessboardcolors{red!15}{blue!15}
   \Body
     1 & 1 & 1  & 1  & 1  & 1 & 1 & 1 \\
     1 & 2 & 3  & 4  & 5  & 6 & 7 \\
     1 & 3 & 6  & 10 & 15 & 21 \\
     1 & 4 & 10 & 20 & 35 \\
     1 & 5 & 15 & 35 \\
     1 & 6 & 21 \\
     1 & 7 \\
     1
   \end{NiceTabular}}
```





Of course, if you have to draw many diagrams like this one, you should consider using a dedicated package, such as atableau.

History

The development of the package nicematrix is done in the following GitHub depot:
`https://github.com/fpantigny/nicematrix`

The successive versions of the file `nicematrix.sty` provided by TeX Live are also available on the SVN server of TeX Live:

`https:www.tug.org/svn/texlive/trunk/Master/texmf-dist/tex/latex/nicematrix/nicematrix.sty`

Changes between version 7.12 and 7.13

New key `overlap`
New key `xdots/number-of-dots`
Special value `row-height` for `columns-width`

Changes between version 7.11 and 7.12

New key `color-of-false` for the color of the backgrounds of the rows and columns created by the command `\FalseRow` and the letter `F`.
New key `empty` for the command `\Block`: all the cells of the block will be considered as empty (that's meaningful, for example, for the computation of the "empty corners").

Changes between version 7.10 and 7.11

New keys `xoffset` and `yoffset` in the key `tikz` of the command `\Block` (previously, there was only `offset` which is now a kind of conjunction of `xoffset` and `yoffset`).
New command `\FalseRow` and letter `F` (for a "false column").

Changes between version 7.9 and 7.10

A generalisation of the key `corners` has been added with the values `N`, `S`, `W` and `E` (*north*, *south*, *west* and *east*).

Changes between version 7.8 and 7.9

New keys `default-line` and `rules/fix-vertex`

Changes between version 7.7 and 7.8

New key `draw-trees-in-col` (and keys `trees/color`, `trees/line-width` and `trees/rounded-corners`)

Changes between version 7.6 and 7.7

New commands `\hdashedline`, `\cdashedline` and the letter `;`
New key `notes/no-print` and new command `\NiceTabularNotes`

Changes between version 7.5 and 7.6

New key `create-blocks-in-col`

An "additive syntax" has been added for several keys: `cell-space-limits`, `xdots/brace-shift`, `xdots/shorten`, `notes/code-before`, `code-for-first-row`, etc.
That means that it's possible to write, for example : `cell-space-limits += 3pt`.

Changes between version 7.4 and 7.5

New key `brace-shift` for the commands `\Hbrace` and `\Vbrace`.

Changes between version 7.3 and 7.4

Negative values are now allowed for the keys `hlines` and `vlines`.

Changes between version 7.2 and 7.3

Simplification of the code by using the latest version of LaTeX : 2025-06-01

Changes between version 7.1 and 7.2

A new key `V` is available in the columns of type `X`.

Changes between version 7.0 and 7.1

New commands `\Hbrace` and `\Vbrace`.
New commands `\EmptyColumn` and `\EmptyRow` in the `\CodeBefore`.

Changes between version 6.29 and 7.0

The package nicematrix is now compatible (for the tabulars) with the Tagging Project.

Changes between version 6.28 and 6.29

Modification in order to be compatible with the next version of array (v. 2.6g).

Changes between version 6.27 and 6.28

Sub-blocks with the character `&` (when the key `ampersand-in-block` is in force).
Keys `p` and `j` for the command `\Block`.
PGF nodes i.1, i.2, i.3, etc.

Changes between version 6.26 and 6.27

New key `light-syntax-expanded`.

Changes between version 6.25 and 6.26

Special color `nocolor`.

Changes between version 6.24 and 6.25

Correction of several bugs.
An instruction `\rowlistcolors` in the tabular stops the effect of a previous `\rowlistcolors`.

Changes between version 6.23 and 6.24

New command `\TikzEveryCell` available in the `\CodeAfter` and the `\CodeBefore` to apply the same TikZ instruction to all the cells and blocks of the array.
New key `offset` in the key `tikz` of a command `\Block`.

Changes between version 6.22 and 6.23

The specifier "`|`" in the preambles of the environments has now an optional argument.
Correction of several bugs.

Changes between version 6.21 and 6.22

Key `opacity` for the command `\Block`.
It's now possible to put a label on a "continuous dotted line" with the specifier "`:`".

Changes between version 6.20a and 6.21

Key `c` for the command `\tabularnote`.
New commands `\rowcolors` and `\rowlistcolors` available in the array itself (previously, there were only two commands `\rowcolors` and `\rowlistcolors` available in the `\CodeBefore`.

Changes between version 6.20 and 6.20a

The value of the key `name` of an environment of nicematrix is expanded (in the TeX sens).
The labels of the tabular notes (created by the command `\tabularnote`) are now composed in an overlapping position *only* when the horizontal alignment mode of the column is `c` or `r`.

Changes between version 6.19 and 6.20

New key `xdots/horizontal-labels`.

Changes between version 6.18 and 6.19

The command `\tabularnote` now supports an optional argument (between square brackets) to change the symbol of the reference of the note.

Changes between version 6.17 and 6.18

New key `opacity` in the commands to color cells, rows and columns.
New key `rounded-corners` for a whole tabular.

Changes between version 6.16 and 6.17

New PGF/TikZ style `nicematrix/cell-node`.
New key `pgf-node-code`

Changes between version 6.15 and 6.16

It's now possible to put any LaTeX extensible delimiter (`\lgroup`, `\langle`, etc.) in the preamble of an environment with preamble (such as `{NiceArray}`) by prefixing them by `\left` and `\right`.
New key `code` for the command `\SubMatrix` in the `\CodeAfter`.

Changes between version 6.14 and 6.15

New key `transparent` for the command `\Block` (with that key, the rules are drawn within the block).

Changes between version 6.13 and 6.14

New keys for the command `\Block` for the vertical position of the content of that block.

Changes between version 6.12 and 6.13

New environment `{TabularNote}` in `{NiceTabular}` with the same semantic as the key `tabularnote` (for legibility).
The command `\Hline` nows accepts options (between square brackets).

Changes between version 6.11 and 6.12

New keys `caption`, `short-caption` and `label` in the environment `{NiceTabular}`.
In `{NiceTabular}`, a caption specified by the key `caption` is wrapped to the width of the tabular.
Correction of a bug: it's now possible to use `\OverBrace` and `\UnderBrace` with unicode-math (with XeLaTeX or LuaLaTeX).

Changes between version 6.10 and 6.11

New key `matrix/columns-type` to specify the type of columns of the matrices.
New key `ccommand` in `custom-line` and new command `\cdottedline`.

Changes between version 6.9 and 6.10

New keys `xdots/shorten-start` and `xdots/shorten-end`.
It's possible to use `\line` in the `\CodeAfter` between two blocks (and not only two cells).

Changes between version 6.8 and 6.9

New keys `xdots/radius` and `xdots/inter` for customisation of the continuous dotted lines.
New command `\ShowCellNames` available in the `\CodeBefore` and in the `\CodeAfter`.

Changes between version 6.7 and 6.8

In the notes of a tabular (with the command `\tabularnote`), the duplicates are now detected: when several commands `\tabularnote` are used with the same argument, only one note is created at the end of the tabular (but all the labels are present, of course).

Changes between version 6.6 and 6.7

Key `color` for `\OverBrace` and `\UnderBrace` in the `\CodeAfter`
Key `tikz` in the key `borders` of a command `\Block`

Changes between version 6.5 and 6.6

Keys `tikz` and `width` in `custom-line`.

Changes between versions 6.4 and 6.5

Key `custom-line` in `\NiceMatrixOptions`.
Key `respect-arraystretch`.

Changes between versions 6.3 and 6.4

New commands `\UnderBrace` and `\OverBrace` in the `\CodeAfter`.
Correction of a bug of the key `baseline` (cf. question 623258 on TeX StackExchange).
Correction of a bug with the columns `V` of varwidth.
Correction of a bug: the use of `\hdottedline` and `:` in the preamble of the array (or another letter specified by `letter-for-dotted-lines`) was incompatible with the key `xdots/line-style`.

Changes between versions 6.2 and 6.3

Keys `nb-rows`, `rowcolor` and `bold` for the command `\RowStyle`
Key `name` for the command `\Block`.
Support for the columns `V` of varwidth.

Changes between versions 6.1 and 6.2

Better compatibility with the classes revtex4-1 (deprecated) and revtex4-2.
Key `vlines-in-sub-matrix`.

Changes between versions 6.0 and 6.1

Better computation of the widths of the `X` columns.
Key `\color` for the command `\RowStyle`.

Index

Symbols

&-in-blocks, 12

A

ampersand-in-blocks, 12
\arraycolor (command of \CodeBefore), 25
\arrayrulecolor, 15
\arrayrulewidth, 15
auto-columns-width
    (clé de {NiceMatrixBlock}), *81*
    (key of {NiceMatrixBlock}), 34
\AutoNiceMatrix, 63

B

baseline (key for an environment), 5
\BAutoNiceMatrix, 63
\bAutoNiceMatrix, 63
blkarray (package), 45
\Block, 6
Blocks in the tabulars, 6–13
\Body, *see* \CodeBefore
bold (key of \RowStyle), 32
booktabs (package), 14
borders (key of \Block), 8
bottomrule (subkey of "notes"), 55
brace (nicematrix/brace is a TikZ style
        used by \Hbrace et \Vbrace), 44
brace-shift (key of \Hbrace and \Vbrace), 44

C

Captions of the tabulars, 53
caption (key of {NiceTabular}), 53
caption-above, 53
ccommand (key of "custom-line"), 19
\cdashedline, 21
\Cdots, 38
\cdottedline, 21
cell-space-bottom-limit, 4, 32
cell-space-limits, 4, 32
cell-space-top-limit, 4, 32
\cellcolor
    command in tabular, 30
    command of \CodeBefore, 25
cellspace (package), 4
\chessboardcolors
    (commande du \CodeBefore), 25, *87*
\cline (LaTeX command), 15
code (key of \SubMatrix), 49
code-after, 46
code-before
    key for an environment, 25
    subkey of "notes", 55
code-for-first-col, 37
code-for-first-row, 37, *61*, *79*
code-for-last-col, 37, *61*, *79*
code-for-last-row, 37, *61*
\CodeAfter, 46–52, *86*
\CodeBefore...\Body, 25, *86, 87*
color
    for the delimiters of the matrices, 63
    key for dotted rules, 42
    key of \OverBrace and \UnderBrace, 51
    key of \RowStyle, 32
    key of "custom-line", 20
color-of-false, 24
colortbl (package), 25
Colour
    of the cells, 25
    of the delimiters of the matrices, 63
    of the rules, 15
cols (key of \rowcolors of \CodeBefore), 27
\columncolor
    command in the preamble of an environment,
        30
    command of \CodeBefore, 25, *75*
columns-type (key of {NiceMatrix}, etc.), 60
columns-width, 33
command (key of "custom-line"), 19
corners (key of an environment), 17, *29*
Corners (rounded —)
    for a block, 7
    for a tabular, 59
Corners (the empty —), 17, *87*
create-blocks-in-col, 13
create-cell-nodes (key of \CodeBefore), 67, *86*
create-extra-nodes, 66
create-large-nodes, 66
create-medium-nodes, 66
\crossbox (defined in an example), *70*
custom-line, 19–23

D

\Ddots, **38**, 71, *78*
default-line, 24
\definecolorseries (command of xcolor), 28
delimiters
    —/color for an environment, 63
    —/color pour \SubMatrix, 49
    —/max-width, 82
delimiters in the preambles, 45
detect-duplicates (subkey of "notes"), 55
\diagbox, 18
Dotted leaders, *78*
Dotted lines, 38–45
dotted (clé de «custom-line»), 21
draw (key of \Block), 7, *85*
draw-trees-in-col, 58

E

empty (key of \Block), 8
empty (key of \TikzEveryCell), 51
\EmptyColumn (command of \CodeBefore), 30
\EmptyRow (command of \CodeBefore), 30
end (key for the rules), 22
end-of-row (to be used with light-syntax), 63
enumitem (package required to use \tabularnote), 53, *74*
enumitem-keys (subkey of "notes"), 55, *74*
enumitem-keys-para (subkey of "notes"), 55
exterior-arraycolsep, 72
extra-height (key of \SubMatrix), 49
extra-left-margin, 67
extra-right-margin, 67

F

F (letter in the preambles), 24
\FalseRow, 24
fill
  key of \Block, 7
  key of \RowStyle, 32
first-col, 36
first-row, 36, *61*
fix-vertex (subkey of rules), 23
footnote (package), 53
footnote (key), 53
footnotehyper (package), 53
footnotehyper (key), 53

H

\Hbrace, 44
\hdashedline, 21
\Hdotsfor, 40
\hdottedline, 21
hhline (package), *77*
\Hline, 15
hlines, *see* Rules
  key for an environment, 16
  key of \Block, 7
  key of \SubMatrix, 49
hlines-except-borders, 17
horizontal-label(s) (key for dotted rules), 42
\Hspace, 39
hvlines, *see* Rules
  key for an environment, 17
  key of \Block, 7
  key of \SubMatrix, 49
hvlines-except-borders, 17, *76*

I

\Iddots, **38**, 71
Incompatibilities, 73
inter (key for dotted rules), 42
iRow (LaTeX counter), 62

J

jCol (LaTeX counter), 62

L

label (key of {NiceTabular}), 53
label-in-list (subkey of "notes"), 55
label-in-tabular (subkey of "notes"), 55
last-col, 36, *61*, *79*
last-row, 36, *61*
\Ldots, 38
\left : used by nicematrix for delimiters in the preambles, 45
left-margin, 66
left-shorten (key of \OverBrace and \UnderBrace), 51
left-xshift (key of \SubMatrix), 48
letter (key of "custom-line"), 19
light-syntax, 63
light-syntax-expanded, 63
\line (command of \CodeAfter), 47
line-style (key for dotted rules), 42
line-style (key for the dotted rules), *79*
Lines in the tabulars, *see* Rules

M

mathdots (package), 38
max-width (subkey of "delimiters"), 82
multiplicity (key of "custom-line"), 20

N

name
  key for an environment, 64
  key of \Block, 8
  key of \SubMatrix, 69
nb-rows (key of \RowStyle), 32
{NiceArrayWithDelims}, 64
{NiceMatrixBlock}, 34, *81*
\NiceMatrixOptions, 1
\NiceTabularNotes, 57
{NiceTabularX}, 35
no-cell-nodes, 64
no-print (subkey of "notes"), 55
nocolor, 32
Nodes of PGF/TikZ, 64–69
non empty (key of \TikzEveryCell), 51
Notes in the tabulars, 53–57, *74*
\NotEmpty, 72
notes (key to customize the notes of a tabular), *74*
nullify (key for dotted rules), 42
nullify-dots, 39
number-of-dots (key for dotted rules), 42

O

\OnlyMainNiceMatrix, 64
opacity
  key of \Block, 7
  key of \RowStyle, 32
  key of commands such as \rowcolor, etc., 26
\OverBrace (command of \CodeAfter and \CodeBefore), 50

overlap (key), 47

P

para (subkey of "notes"), 55
parallelize-diags, 71
\pAutoNiceMatrix, 63
pgf-node-code, 66, *87*

R

radius (key for dotted rules), 42
\rectanglecolor (in \CodeBefore), 25
renew-dots, 41
renew-matrix, 41
\resetcolorseries (command of xcolor), 28
respect-arraystretch (key of \Block), 8
respect-blocks (key of \rowcolors du \CodeBefore), 27
restart (key of \rowcolors of \CodeBefore), 27
\right : used by nicematrix for delimiters in the preambles, 45
right-margin, 66
right-shorten (key of \OverBrace and \UnderBrace), 51
right-xshift (key of \SubMatrix), 48
\rotate, *29*, *32*, 61
rounded-corners
  key of \Block, 7
  key of \RowStyle, 32
  key of {NiceTabular}, 59
row-height (value for columns-width, 34
\rowcolor
  command in tabular, 30, *85*
  command of \CodeBefore, 25, *76*
rowcolor (key of \RowStyle), 32
\rowcolors (command of \CodeBefore), 25
\rowlistcolors (command of \CodeBefore), 25
\RowStyle, 32
Rules in the tabulars, 14–24
rules (key for an environment), 15, *76*
rules/color (key of \Block), 7
rules/width (key of \Block), 7

S

S (the columns S of siunitx), *30*, 60
sep-color (key of "custom-line"), 20
short-caption, 53
shorten (key for dotted rules), 42
shorten-end (key for dotted rules), 42
shorten-start (key for dotted rules), 42
\ShowCellNames (command of \CodeAfter and \CodeBefore), 60
siunitx (package), 60
slim (key of \SubMatrix), 49
small (key for an environment), 62
{smallmatrix} (environment of amsmath), 62
standard-cline, 15
start (key for the rules), 22
style (subkey of "notes"), 55, *74*
sub-matrix (key of \CodeAfter, with subkeys), 46
\SubMatrix (command of \CodeAfter and \CodeBefore), **48**, 69, *83*, *86*

T

\tabularnote, 53, *74*
{TabularNote}, 54
tabularnote (key of {NiceTabular}), 54
tabularx (package), 35
Tagging Project, 73
tcolorbox (package), *76*
tikz-braces (key at load-tome of nicematrix), 44
TikZ (use with nicematrix), 64
\TikzEveryCell (command of \CodeAfter and \CodeBefore), **51**
tikz
  key of \Block, 8, *75*
  key of "borders" de \Block, 8, *78*
  key of "custom-line", 22
total-width (key of "custom-line"), 22
transparent (key of \Block), 8, *75*
trees (plusieurs sous-clés), 58

U

\UnderBrace (command of \CodeAfter and \CodeBefore), 50

V

V (the columns V of varwidth), 34, 66
v-center (key of \Block), 11
varwidth (package), 34, 66
\VAutoNiceMatrix, 63
\vAutoNiceMatrix, 63
\Vbrace, 44
\Vdots, 38
\Vdotsfor, 40
vlines, *see* Rules
  key for an environment, 16
  key of \Block, 7
  key of \SubMatrix, 49
vlines-in-sub-matrix, 84

W

width
  key of {NiceTabular}, 35
  subkey of "rules", 15
Width of the columns, 33–36
width-of-false, 24

X

X (the columns X), 35
xdots (and its subkeys), 38
xshift (key of \SubMatrix), 48

Y

yshift (key of \OverBrace and \UnderBrace), 51