Manual for `sgamex.sty`

by
Martin J. Osborne
martin.j.osborne@gmail.com

For Version 1.0, 2020-2-7

1. Introduction

`sgamex.sty` is a LaTeX style file for drawing strategic games. The latest version is available at
https://www.economics.utoronto.ca/osborne/latex.
The style reimplements the `game` environment from `sgame.sty` to avoid incompatibilities with other packages, in particular `array.sty`. (It does not currently implement the `game*` environment in `sgame.sty`.)

One small inconvenience remains: because of a clash between the `hhline` and `beamer` packages, if you are using `sgamex` with `beamer` you need to add the following lines just before `\begin{document}`:

```
\makeatletter
\let\zz\reset@color
\def\reset@color{\kern\z@\zz}
\makeatother
```

(See this page for details.)

The core of `sgamex.sty` was written by Enrico Gregorio (`egreg`) in response to a question on StackExchange. The method of saving and restoring the value of `\arrayrulecolor` is due to David Carlisle (see his answer on this page). I am grateful to Enrico and David for their code.

The only reason to use `sgame.sty` or its cousin `sgamevar.sty` rather than `sgamex.sty` is for the `game*` environment.

Please notify me of bugs.

2. Installation

- Put `sgamex.sty` in a directory from which TeX reads input files. (For MiKTeX the directory might be something like `\miktex\localtexmf\tex\latex\` or a subdirectory thereof.)
- Let TeX know that `sgamex` has arrived. (In MiKTeX, "refresh the filename database".)

- To use the package in a document, put the line `\usepackage{sgamex}` in the preamble. It loads the `xcolor` package with the option `table`, so if your document loads another package that loads `xcolor` without an option, you need to load `sgamex` first or to load `xcolor` early with the `table` option.

3. Description

The style defines the environment `game`, which can be used in four ways, depending on the values of its optional arguments.

3.1 No optional argument

Without any optional arguments, its syntax is

`\begin{game}{`*number-of-rows*`}{`*number-of-columns*`}`*game-body*`\end{game}`

where *number-of-rows* is the number of rows in the game and *number-of-columns* is the number of columns. *game-body* consists of rows in the style of the rows in a `tabular` environment, `&` "tabbing" to a new cell and `\\` ending a row.

The following example is illustrated in Figure 1.

```
\def\sgtextcolor{blue}%
\def\sglinecolor{red}%
\begin{figure}[htb]\hspace*{\fill}%
\begin{game}{2}{2}
        & \(L\)    & \(R\)\\
\(T\)   &\(2,2\)   &\(100,35\)\\
\(B\)   &\(3,0\)   &\(350,16\)
\end{game}
\hspace*{\fill}%
\caption[]{A two row, two column strategic game.}
\end{figure}
```

	L	R
T	$2,2$	$100,35$
B	$3,0$	$350,16$

Figure 1. A two row, two column strategic game.

- Note that the width of every column is the same and the content of each cell of the table is centered vertically within the cell. (Regarding the latter, see section 5.)
- As in a LaTeX `tabular` environment, spaces on each side of the text in each cell in the input file are ignored. (Thus in the `tex` file you can, if you wish, align the columns using spaces or tabs, as I have done.)
- If all the text in every cell of your payoff matrix is in math mode, you can avoid typing all the `\(`'s and `\)`'s by setting `\gamemathtrue`, either by putting this command in your file, or by changing `\gamemathfalse` to `\gamemathtrue` in the style file. Under this setting, all text in every cell of the payoff matrix is automatically in math mode. (Note that even if the text in the cells consists only of comma-separated pairs of numbers, it should still be in math mode, otherwise the spacing is not correct.)

3.2 One optional argument: game label

With one optional argument, the syntax of the `game` environment is

`\begin{game}{`*number-of-rows*`}{`*number-of-columns*`}[`*game-label*`]`*game-body*
`\end{game}`

where *number-of-rows* is the number of rows in the game, *number-of-columns* is the number of columns, and *game-label* is a label for the game that is centered beneath the game. (The sense in which it is centered may be changed by setting `\irpawcgltrue` or `\irplwcgltrue`—see section 9.) This case is useful if two or more games are positioned side-by-side, and each requires a label.

```
\begin{figure}[htb]\hspace*{\fill}%
\begin{game}{2}{2}[\(A\)]
        & \(L\)      & \(R\)\\
\(a\)   &\(2,2,1\)   &\(0,3,0\)\\
\(b\)   &\(3,0,2\)   &\(1,1,4\)
\end{game}\hspace*{20mm}%
\begin{game}{2}{2}[\(B\)]
        & \(L\)      & \(R\)\\
\(a\)   &\(2,3,0\)   &\(0,4,1\)\\
\(b\)   &\(3,1,2\)   &\(1,2,0\)
\end{game}\hspace*{\fill}%
```

```
\caption[]{A three player strategic game, in which
player~3 chooses \(A\) or \(B\).}
\end{figure}
```

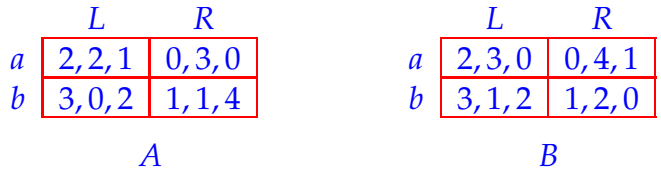

Figure 2. A three player strategic game, in which player 3 chooses *A* or *B*.

3.3 Two optional arguments: player labels

In the case of two optional arguments, the syntax of the `game` environment is

`\begin{game}{`*number-of-rows*`}{`*number-of-columns*`}[`*row-player-label*`]`
`[`*column-player-label*`]`*game-body*`\end{game}`

where *number-of-rows* is the number of rows in the game, *number-of-columns* is the number of columns, and *row-player-label* and *column-player-label* are the names of the players.

```
\begin{figure}[htb]\hspace*{\fill}%
\begin{game}{2}{3}[Player~1][Player~2]
        & \(L\)    & \(M\)    & \(R\)\\
\(T\)   &\(2,2\)   &\(2,0\)   &\(0,3\)\\
\(B\)   &\(3,0\)   &\(0,9\)   &\(1,1\)
\end{game}\hspace*{\fill}%
\caption[]{A two row, three column strategic game with
player labels.}
\end{figure}
```

See Figure 3 for the result.

3.4 Three optional arguments: player labels and game label

The case of three optional arguments combines the two previous cases: the syntax of the `game` environment is

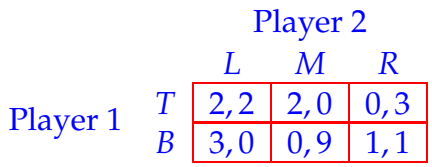

Figure 3. A two row, three column strategic game with player labels.

`\begin{game}{`*number-of-rows*`}{`*number-of-columns*`}[`*row-player-label*`]`
`[`*column-player-label*`] [`*game-label*`]`*game-body*`\end{game}`

where *number-of-rows* is the number of rows in the game, *number-of-columns* is the number of columns, *row-player-label* and *column-player-label* are the names of the players, and *game-label* is a label for the game.

```
\begin{figure}[htb]\hspace*{\fill}%
\begin{game}{2}{3}[Player~1][Player~2][\(A\)]
        & \(L\)    & \(M\)    & \(R\)\\
\(T\)   &\(2,2\)   &\(2,0\)   &\(0,3\)\\
\(B\)   &\(3,0\)   &\(0,9\)   &\(1,1\)
\end{game}\hspace*{\fill}%
\caption[]{A two row, three column strategic game with
player labels and a label.}
\end{figure}
```

See Figure 4 for the result.

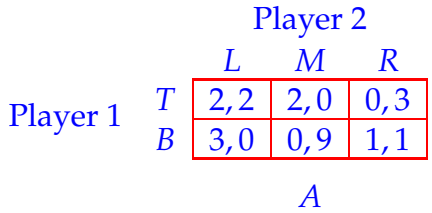

Figure 4. A two row, three column strategic game with player labels and a label.

4. Cell formatting

You may regard the format of the payoffs in the first column of the game in Figure 5 as unsatisfactory, and the formatting in the left panel of Figure 6 as superior. This latter formatting is implemented by adding "phantom" minus signs before the "2" in the top left cell and before the "3" in the bottom

left cell. The top left cell, for example, is entered as `\(-2,\phantom{-}2\)`. (`\phantom{<x>}` leaves a space whose size is equal to the size of the object `<x>`.)

	L	R
T	$-2,2$	$0,3$
B	$3,-4$	$1,1$

Figure 5. A strategic game.

You may want to consider other formats. The right panel of Figure 6 shows an alternative. This format is achieved be entering the element of each cell as

```
\begin{array}{c}#1\\#2\end{array},
```

where `#1` is player 1's payoff and `#2` is player 2's payoff. (Actually, I defined the macro

```
\def\stackedpayoffs#1#2{%
  \begin{array}{c}#1\\#2\end{array}
}
```

and entered each element as `\stackedpayoffs{#1}{#2}`. I set `\gamestretch` to be 2.1 to improve the appearance of the game.)

	L	R
T	$-2,\ \ 2$	$0,3$
B	$3,-4$	$1,1$

	L	R
T	-2 2	0 3
B	3 -4	1 1

Figure 6. Two alternative presentations of the game in Figure 5.

Figure 7 shows two more alternatives. The one on the left is achieved by entering the element of each cell as

```
\begin{array}{rl}#1&\\&#2\end{array},
```

and the one on the left differs only in that there is a `\phantom{-}` in front of player 2's payoff in the top left cell, player 1's payoff in the bottom left cell, and player 2's payoffs in both of the right cells.

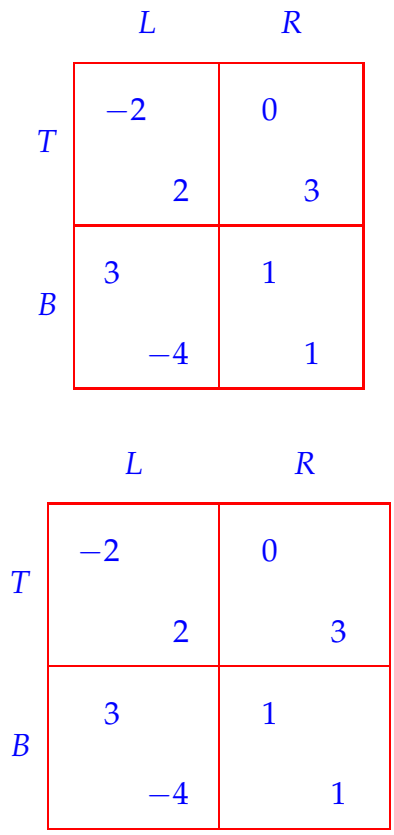

Figure 7. Two additional presentations of the game in Figure 5.

5. Vertical position of text within cells

The vertical distance between the lines at the top and bottom of a row in LaTeX's `tabular` environment is equal to `\arraystretch` times `\baselineskip`. The baseline of the text in each cell is placed 30% of the distance from the bottom line. Thus if `\arraystretch` is large relative to the height of the text, the text is not near the middle of the row. (See the left panel of Figure 8; or try setting `\arraystretch` equal to 5 and process a document with a `tabular` environment.) (See the definitions of `\strutbox` and of `\@array` in `latex.ltx`.)

sgame.sty makes the vertical distance between the lines at the top and bottom of a row of a game the same (\arraystretch times \baselineskip), but places the baseline of the text in each cell so that the distances above and below an upper case I placed on the baseline are equal. (See the right panel of Figure 8.) (If you'd like to use another character, define this character to be \sg@alignchar.)

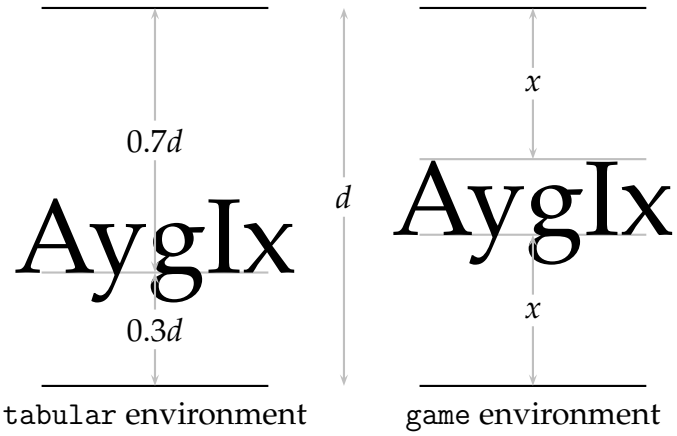

Figure 8. Vertical positioning of text within cells in the tabular and game environments.

If the height of the text in a cell exceeds \arraystretch times \baselineskip, then the height of the row is increased to accommodate the text (just as the distance between lines of text is increased to accommodate large text). The result is that the lines at the top and bottom of the row touch the top and bottom of the text, and this row is taller than the others. To improve the appearance of the table you may want to increase \gamestretch (analogous to \arraystretch), to put some white space above and below your oversized text.

6. Thick lines

To change the width of the cell boundaries, set \sgrulewidth (by writing something like \sgrulewidth2mm or, if you prefer, \setlength{\sgrulewidth}{2mm}). In addition, you may need to stretch the game vertically to accommodate the thicker lines. To do so, set \gamestretch (by writing something like \renewcommand{\gamestretch}{2}). Here's an example.

```
\def\sgtextcolor{blue}%
```

```
\def\sglinecolor{red}%
\renewcommand{\gamestretch}{2}
\sgrulewidth2mm
\begin{figure}[htb]\hspace*{\fill}%
\begin{game}{2}{2}
        & \(L\)    & \(R\)\\
\(T\)   &\(2,2\)   &\(100,35\)\\
\(B\)   &\(3,0\)   &\(350,16\)
\end{game}\hspace*{\fill}%
\caption[]{Thick lines.}
\end{figure}
```

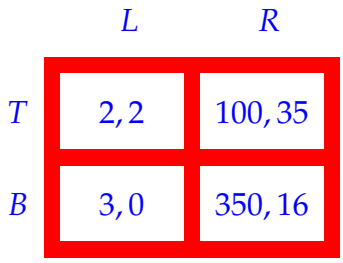

Figure 9. Thick lines.

7. Crossing out actions

To create Figure 10, add `\usepackage{pstricks,pst-node}` to the preamble of your document, define a couple of macros specifying what the strike-out lines should look like, add some node labels to the game, and specify how the nodes should be connected. The only difficulty comes in getting the lengths of the lines right. I have been unable to get the parameters to work as they are described in Tim van Zandt's manual, but with the help of BaKoMa Word it's easy to fiddle around with them until the lines are the way you want them. The code for this game follows the figure.

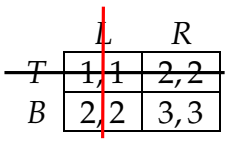

Figure 10. Crossing out actions in a strategic game.

```
\def\sgtextcolor{black}%
\def\sglinecolor{black}%
% define the strikeout lines
\newcommand\strike[2]{%
  \ncline[linewidth=1.2pt,nodesep=8pt]{#1}{#2}}
\newcommand\redStrike[2]{%
  \ncline[linewidth=1.2pt,nodesep=-14pt,linecolor=red]{#1}{#2}}

\begin{game}{2}{2}
                            &\rnode[t]{a12}{\(L\)}   &\(R\)\\
\Rnode[href=20]{a21}{\(T\)} &\(1,1\)                 &\Rnode{a23}{\(2,2\)}\\
\(B\)                       &\rnode[b]{a32}{\(2,2\)} &\(3,3\)
\end{game}
% specify the nodes to be connected
\strike{a21}{a23}
\redStrike{a12}{a32}
```

8. Coloring cells

The style uses the `colortbl` package, so you can use any of the features that `colortbl` provides. For example, you can color a cell by adding `\cellcolor{<color>}` before its contents, as in the following example; the output is in Figure 11.

```
\renewcommand{\gamestretch}{1.5}

\begin{figure}[htb]\hspace*{\fill}%
\begin{game}{2}{2}
&\(L\)&\(R\)\\
\(T\)&\cellcolor{yellow}\(1,1\)&\(2,2\)\\
\(B\)&\(3,0\)&\(0,3\)\\
\end{game}\hspace*{\fill}%
\caption[]{A game with a colored cell.}\label{f:coloredCell}
\end{figure}
```

9. Parameters

`\gamestretch`: a number that controls the spacing between rows, like `\arraystretch` for tables. Possible values: positive real numbers. Default:

	L	R
T	$1,1$	$2,2$
B	$3,0$	$0,3$

Figure 11. A game with a colored cell.

value of `\arraystretch` when `sgame.sty` is loaded (1 in standard styles). Example: `\renewcommand{\gamestretch}{2}`.

`\sgcolsep`: horizontal padding within cells. Possible values: any dimension. Default: value of `\tabcolsep` when `sgame.sty` is loaded. Example: `\sgcolsep=10pt`.

`\sglabelsep`: vertical space between game and label. Possible values: any dimension. Default: 5pt. Example: `\sglabelsep=10pt`.

`\irpawcgltrue` ("include row player actions when centering game label"): causes game label to be centered under box consisting of row player actions and payoff matrix. Default: `\irpawcglfalse` (i.e. game label is centered under box consisting only of payoff matrix). Example: `\irpawcgltrue`.

`\irplwcgltrue` ("include row player label when centering game label"): causes game label to be centered under box consisting of row player label, row player actions, and payoff matrix. (Is overridden by `\irpawcgltrue`.) Default: `\irplwcglfalse` (i.e. game label is centered under payoff matrix). Example: `\irplwcgltrue`.

`\gamemathtrue`: put all text in cells in payoff matrix in math mode. Default: `\gamemathfalse`. Example: `\gamemathtrue`.

`\sglinecolor`: color of lines around payoffs. Default: `black`. Example: `\def\sglinecolor{lightgray}`.

`\sgtextcolor`: color of text in game (action labels, payoffs, player labels). Default: `black`. Example: `\def\sgtextcolor{blue}`.

10. History

Version 1

2020/2/7 First version (reimplementation of `game` environment in `sgame.sty`, the first version of which was written in December 1993).