The `genyoungtabtikz` package, version 1.10

Matthew Fayers
Queen Mary University of London

April 13, 2015

Abstract

A LaTeX package to produce Young diagrams and tableaux. It is designed to have the economy of syntax of the `youngtab` package, but with many additional features and the graphical power of PGF/Tikz.

Contents

1 Introduction

The `youngtab` package by Volker Börchers and Stefan Gieseke provides a very easy way to typeset Young diagrams and Young tableaux using the basic box-drawing capabilities of TeX. The purpose of this package, which is inspired by `youngtab`, is to increase the functionality and to improve some of the behaviour using the advanced drawing capabilities of PGF/Tikz. Much of the syntax we employ is taken directly from `youngtab`, so `youngtab` users should find this package easy to use.

One feature from `youngtab` which is missing here is the facility for skew diagrams. These can be achieved with the `\gyoung()` command (see below), but we do not explicitly implement them.

`genyoungtabtikz` calls the `tikz` package, so you need to have this installed, but you don't need to know how to use it. To use `genyoungtabtikz`, put the file `genyoungtabtikz.sty` somewhere LaTeX will find it (e.g. in a `texmf` tree or the same directory as your document) and call it with `\usepackage{genyoungtabtikz}`, possibly with some of the options described below.

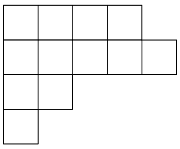
2 Commands

The package implements several commands for drawing different kinds of diagrams (along with some options for fine-tuning). These diagram-drawing commands can be used in either math-mode or text-mode (and there are also versions that can be used within `tikz` diagrams).

2.1 Young diagrams

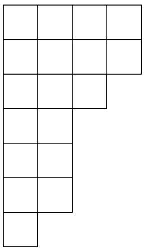
As in `youngtab`, the command `\yng()` is used to typeset a simple Young diagram. The argument between the brackets consists of the row lengths (which should be positive integers, but need not be decreasing) separated by commas.

```
\yng(4,5,2,1)
```



Additionally, `genyoungtabtikz` supports grouping together consecutive equal parts with a ˆ (which is supposed to evoke the superscript conventionally used for writing partitions).

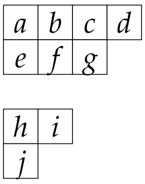
```
\yng(4^2,3,2^3,1)
```



2.2 Young tableaux

As in `youngtab`, the command `\young()` is used to typeset a simple Young tableau. The argument consists of strings of characters separated by commas; each string corresponds to a row of the tableau, with one character in each box.

```
\young(abcd,efg,,hi,j)
```



Observe that (unlike in `youngtab`) rows are not required to be non-empty.

If you want to put more than one character in a box, then you have two options:

- as in `youngtab`, you can just define a command which expands to the characters you want;
- (new in v1.8) you can enclose the characters in <>.

```
\newcommand\fourteen{14}
\newcommand\sixteen{2^4}
\young(4567,\fourteen<15>\sixteen<17>)
```

4	5	6	7
14	15	2^4	17

You also need to take special action if you want to use one of the characters which is special to LaTeX (such as \ or &) or one of the characters which is special in the `\young()` environment, i.e. `,` or `)` or `<` or `>` or `!` (about which more later). This also applies to `.` (which has special meaning within the code for `\young()` and to a space. For these characters, you essentially have the two options above. However, for `)` and `.` you have to use the first option (defining a command) above, and this also applies if you want to put several characters in a cell, one of which is >. For `!`, you have to use the second option (i.e. `<!>`), though I can't really figure out why the first option doesn't work.

```
\newcommand\cbr)
\newcommand\tmp{1^>}
\young(<,>\cbr<<><!>,>\ \tmp,\backslash)
```

,	)	<	!
>		$1^>$	
\			

2.3 *e*-residue diagrams

Often used in modular representation theory and symmetric function theory is the *e-residue diagram*: this is a Young tableau in which the box in row i and column j is filled with the integer $j-i$, possibly reduced modulo some positive integer e. Of course, this can easily (but tediously) be typeset using `\young()`, but here we provide a command `\yngres()` to make this very easy. The syntax is as for `\yng()`, except that the argument is preceded by the non-negative integer e and a comma. If $e = 0$, then the integers just appear as integers.

```
\yngres(3,4^2,1)
\yngres(0,5,2,1,2)
```

0	1	2	0
2	0	1	2
1			

0	1	2	3	4
−1	0			
−2				
−3	−2			

If you want the entries shifted (modulo e) by a constant C, then the command `\Ycorner` (which takes the argument C) will make this change (globally).

```
\Ycorner-1
\yngres(3,4^2,1)
\yngres(0,5,2,1,2)
```

2	0	1	2
1	2	0	1
0			

−1	0	1	2	3
−2	−1			
−3				
−4	−3			

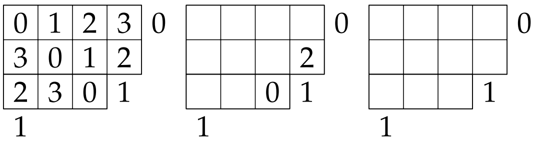
More refinement is possible, in that the residues of *addable nodes* (i.e. places where a box could be added) can be shown, or only the residues of removable boxes can be shown, etc. To control this, there are commands `\Yaddables`, `\Yremovables` and `\Yinternals`, each of which should be followed by a `1` for true or a `0` for false. Note that if you choose not to label the removable boxes, then automatically the non-removable boxes are not labelled either, so there are six possibilities in total.

```
\yngres(4,4^2,3)
\Yinternals0
\yngres(4,4^2,3)
\Yremovables0
\yngres(4,4^2,3)
```

0	1	2	3
3	0	1	2
2	3	0	

			2
		0	


```
\Yinternals1\Yaddables1
\yngres(4,4^2,3)
\Yinternals0
\yngres(4,4^2,3)
\Yremovables0
\yngres(4,4^2,3)
```

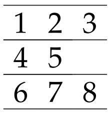


These additional options don't make a lot of sense for compositions which are not partitions, i.e. whose parts are not in decreasing order, so you may get some unexpected results if you try this. Note also that you need to use the ^ notation for consecutive equal parts (because this is how `genyoungtabtikz` determines what the addables and removables are).

2.4 Young tabloids

The *Young tabloid* is used in representation theory, and consists of a Young tableau with the vertical lines removed. This is achieved with the command `\youngtabloid()`, whose syntax is exactly the same as for `\young()`.

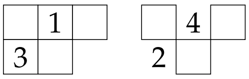
```
\youngtabloid(123,45,678)
```



2.5 Generalised Young diagrams

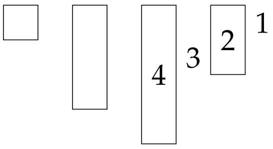
In this section we describe the command `\gyoung()`, which is used for typesetting more complicated diagrams: boxes can be larger, or invisible (but with visible content). As with `\young()`, the argument consists of strings separated by commas. But each string now consists of units corresponding to boxes; each unit begins with one of the symbols `;:|/_^`''. The symbol `;` means a normal box; if this symbol is followed by a character (other than one of `;:|/_^`'',.!`) then this character is put in the box; otherwise the box is left empty. The symbol `:` works in the same way, but the box is not drawn. (The symbol `;` can be omitted if this produces an unambiguous result, i.e. at the start of a row, or following a box with content.) As with `\young()`, the content of a box can be a control sequence which expands to a string of characters, or a string of characters enclosed in <>.

```
\gyoung(;;1;:;:4;,3;:::2;)
```



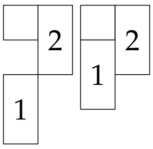
The other symbols `|/_^`''`are used to provide enlarged boxes. The symbol `|` must be followed by a number y, and produces a box y times the normal height (with its top edge at the normal height). Again, if a non-special character follows the number, then this is used as the content of the box. The symbol / is used in exactly the same way, and produces a box which is not drawn.

```
\gyoung(;/2|3/4|44/33|22:1)
```



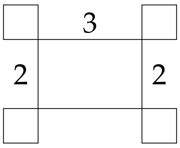
It might appear that the height of an undrawn box with no content is irrelevant, and this is usually the case. However, when drawing boxes of non-standard height and starting a new row of boxes, `\gyoung()` needs to know what vertical shift to apply; this is determined by the height of the *last* box in the upper row. So you can adjust this by adding an empty undrawn box of the appropriate height.

```
\gyoung(;|22,|21)
\gyoung(;|22:,|21)
```



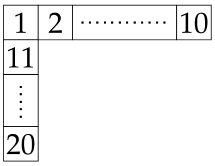
The symbol _, which is followed by a number x, produces a box x times the normal width (possibly with content) and ˆdoes the same for an undrawn box x times the usual width. The symbol ‘, which is followed by x and then y, produces a box with unusual height and width, and ’ does the same for an undrawn box. Note that in any of these cases, x and y need not be positive integers, but if a multi-character number is used, you need to make it a defined command; I haven’t implemented a <> option for this, because multi-character numbers will typically involve the symbol . which wouldn’t work anyway.

```
\gyoung(;ˆ33;,/22‘32/22,;ˆ3;)
```



There are also two special control sequences to be used instead of the content of a box: these are \hdts and \vdts which will fill the box with a horizontal line of dots or a vertical line of dots.

```
\newcommand\sesqui{1.5}
\newcommand\eleven{11}
\gyoung(12_3\hdts<10>,%
\eleven,|\sesqui\vdts,<20>)
```

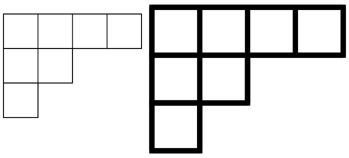


3 Refinement

3.1 Scaling of boxes and line thickness

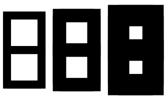
By default, the lines are 0.3pt thick, and the boxes have a side length of 13pt. As in youngtab, these can be changed (globally) using the commands \Ylinethick and \Yboxdim.

```
\yng(4,2,1)
\Ylinethick{2pt}
\Yboxdim{18pt}
\yng(4,2,1)
```



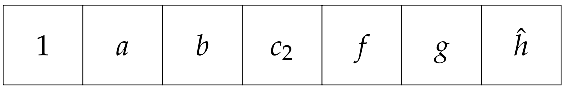
Note that, in contrast to youngtab, the dimension of the box is not just the size of the *interior* of the box – it’s the distance from the middle of one line to the middle of the next.

```
\Ylinethick{3pt}
\yng(1,1)
\Ylinethick{5pt}
\yng(1,1)
\Ylinethick{8pt}
\yng(1,1)
```



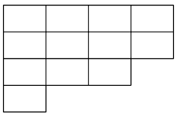
Note also an improvement (in our opinion) on youngtab: when the boxes are scaled, the text remains vertically centred in the boxes – well, sort of. It would not be a great idea to centre the contents of each box, since we want the contents in boxes in the same row to have the same baseline. So the rule is that each character has the same baseline as a vertically centred 1 would have.

```
\Ylinethick{.3pt}
\Yboxdim{30pt}
\young(1ab<c_2>fg<\hat h>)
```



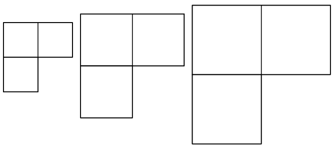
genyoungtabtikz contains a feature not present in youngtab, which is that we allow the horizontal and vertical dimensions of boxes to be changed independently, so that non-square boxes can be produced. This is done with the commands \Yboxdimx and \Yboxdimy.

```
\Yboxdimx{16pt}
\Yboxdimy{10pt}
\yng(4^2,3,1)
```



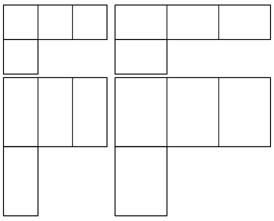
We allow scaling of individual diagrams (without making a global change). For example, the command \yngs() can be used to draw a Young diagram with the boxes scaled by a factor s (which is the argument before the first comma).

```
\Yboxdim{13pt}
\yng(2,1)
\yngs(1.5,2,1)
\yngs(2,2,1)
```



The command \yngx() works in the same way, except that the boxes are only scaled horizontally. \yngy() works similarly, and \yngxy() scales horizontally and vertically according to the first two arguments.

```
\yng(3,1)
\yngx(1.5,3,1)\\
\yngy(2,3,1)
\yngxy(1.5,2,3,1)
```

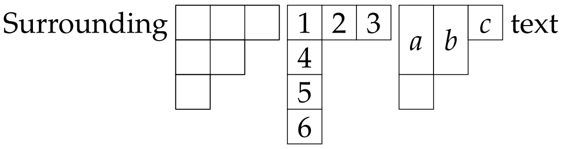


The other commands \young(), \youngtabloid(), \yngres() and \gyoung() all have the same scaling options.

3.2 Vertical alignment

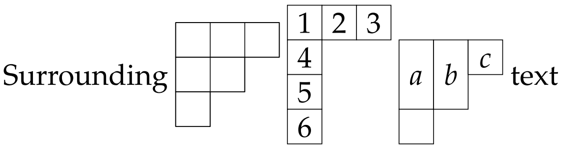
Vertical alignment of diagrams is not handled in the same way as in youngtab. By default the baseline of the text in the standard-height boxes in the first row (or where the text would be, if the boxes are empty or there are no standard-height boxes in the first row) is lined up with the baseline of the surrounding text.

```
Surrounding
\yng(3,2,1)
\young(123,4,5,6)
\gyoung(|2a|2b;c/2,;)
text
```



But as in youngtab, there's a vertical-centering option, activated either with the command \Yvcentermath1 (use \Yvcentermath0 to switch it off again) or by invoking the vcentermath option when calling genyoungtabtikz. Now the baseline of the text in the middle row of boxes (or between the two middle rows, if there are an even number) is lined up with the surrounding text. (The behaviour when there are boxes of non-standard height in `\gyoung()` is more awkward, and some experimenting may be needed.)

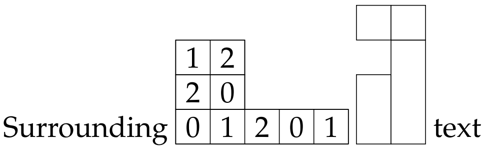
```
\Yvcentermath1
Surrounding
\yng(3,2,1)
\young(123,4,5,6)
\gyoung(|2a|2b;c/2,;)
text
```



3.3 The French and Russian conventions

By default we use the *English* convention for our diagrams, where successive rows of the diagram are lower down the page. But the French convention (where each row is above the previous one) is also implemented: this can be switched on with `\YFrench` or using the `French` package option.

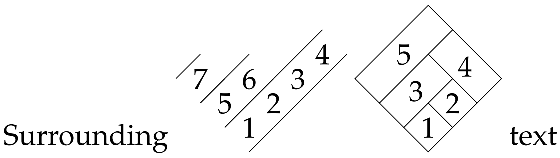
```
\YFrench
Surrounding
\yngres(3,5,2,2)
\gyoung(|2|3,;;)
text
```



Note the behaviour of boxes of non-standard height in `\gyoung`: they now have their *bottom* edge at the usual height.

We also implement the *Russian* convention, which is just the French convention rotated 45° anti-clockwise. Again, there is a command `\YRussian` and a package option `Russian`.

```
\YRussian
Surrounding
\youngtabloid(1234,56,7)
\gyoung(;1;2|24:,_23,_35)
text
```

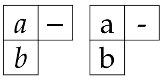


You can return to the English convention with the command `\YEnglish`.

3.4 Math-mode entries in tableaux

As in `youngtab`, you have an option of whether the contents of boxes are math-mode or text-mode, and this is changed with `\Ystdtext` (which is followed by a `0` or `1`). Text-mode can also be switched on with the `stdtext` package option.

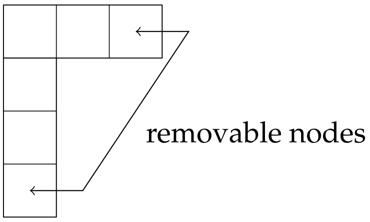
```
\young(a-,b)
\Ystdtext1
\young(a-,b)
```



3.5 Diagrams within `tikzpicture`s

Each time you use one of the drawing commands above, it creates a separate `tikzpicture` environment. If you're a confident user of `tikz` and want to include one of these diagrams in your own `tikzpicture` environment, then you can use the commands `\tyng()`, `\tyoung()` et cetera, which don't create a separate `tikzpicture` environment. The syntax for `\tyng()` is as for `\yng()`, but with two length arguments at the start, which are used as coordinates to place the diagram; this is where the lower-left corner of the first row of the diagram will be placed.

```
\Yboxdim{1cm}
\begin{tikzpicture}[scale=.7]
\tyng(0cm,3cm,3,1^3)
\draw[<->](.5,.5)--(1.5,.5)%
  to node[auto,swap]{removable nodes}%
  (3.5,3.5)--(2.5,3.5);
\end{tikzpicture}
```



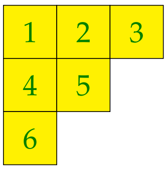
The scaled versions `\yngs()`, `\yngx()`, `\yngy()` and `\yngxy()` do not have counterparts with `\tyng()` ; you can do your own scaling with your `tikzpicture` environment.

The Russian convention won't work here – diagrams will appear in the French convention. I have no plans to fix this – I recommend that you rotate your whole `tikz` diagram use the option `rotate=45`.

3.6 Colour

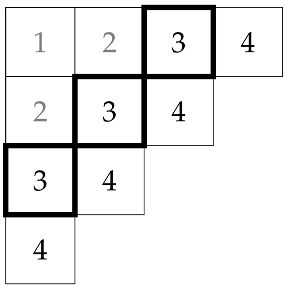
By default, the lines and text are rendered in black, and the boxes are filled in white. You can change these colours using the commands `\Ylinecolour`, `\Ynodecolour` and `\Yfillcolour`, which all take `tikz`-style colour arguments.

```
\Yboxdim{20pt}
\Ylinecolour{blue}
\Ynodecolour{green!50!black}
\Yfillcolour{yellow}
\young(123,45,6)
```



You can also change the opacity of the box-filling with the command `\Yfillopacity`, which takes a number between 0 and 1 as its argument. This can be very useful for overlaying diagrams using the variants `\tyoung` etc.

```
\begin{tikzpicture}[scale=2]
\tyoung(0cm,0cm,1234,234,34,4)
\Yfillopacity{.5}
\tyng(0cm,0cm,2,1)
\Yfillopacity{0}
\Ylinethick{2pt}
\tgyoung(0cm,0cm,::;,:;,;)
\end{tikzpicture}
```



3.7 Making changes mid-diagram

A new feature in v1.7 is the ability to issue LaTeX commands part-way through the argument of `\young()`, `\youngtabloid()` or `\gyoung()`. To do this, include the character `!` followed by any command. If the command is more than one control sequence, then (as with contents of boxes) you can enclose it in <>. For example, you might want to change the fill colour or line thickness part-way through.

```
\newcommand\ylw{\Yfillcolour{yellow}}
\young(12!\ylw3,4!<\Ylinethick{2pt}>56)
```