

The kormap package*

Ben Frank

<https://gitlab.com/benfrank/kormap>

September 16, 2020

This package provides a simple way to typeset Karnaugh maps including automated generation of gray code and options to draw bundles of adjacent cells (implicants).

Contents

1	Introduction	2	4.1	Generating the Gray code	9
2	Drawing Karnaugh maps	3	4.2	Internal code to automate output	10
2.1	Basic commands and environments . . .	3	4.3	Implicant-related code . . .	12
2.2	Drawing Bundles (implicants)	4	4.4	User-interface code	16
2.3	Styling the nodes	4			
				Change History	21
3	Examples	5			
4	Implementation	9			

*This document corresponds to kormap v0.3.5, dated 2020/09/16.

1 Introduction

kvmap aims to provide a *user-friendly* (i.e. less typing) way to typeset Karnaugh maps including the surrounding gray code and bundles of cells (implicants). This package relies on amsmath, xparse (with expl3) and tikz.

Drawing Karnaugh maps is not that uncommon and there are already packages available on CTAN that provide means to typeset them:

- askmaps – This package lets you draw American style Karnaugh maps but restricts you to five variables.
- karnaugh-map – This package allows you to typeset Karnaugh maps with up to 6 variables. Unfortunately, the user has to provide many arguments which makes changing the input error-prone.
- karnaugh – This package lets you typeset Karnaugh maps up to ten variables but I wanted more of what askmaps calls “American style”.
- karnaughmap – This package aims to be a more customizable version of karnaugh, so unfortunately, it does not meet my style requirements either.
- There is also a TikZ library called karnaugh which allows you to use 12 variables (or 4096 cells) and has superior styling options, even for maps with Gray code. But personally I do not like the input format.

So these are basically the reasons, why I need yet another Karnaugh map package. And this package is based on my personal needs, so if you are missing some exciting feature, feel free to open an issue at Gitlab.

Acknowledgements Special thanks to [TeXnician](#) who provided a first version of the Karnaugh map code¹ and [Marcel Krüger](#) who developed the relevant code to generate sequences of gray code².

¹see <https://tex.stackexchange.com/a/425135>

²see <https://tex.stackexchange.com/a/418887>

2 Drawing Karnaugh maps

2.1 Basic commands and environments

<code>\kvmsetup</code>	<code>\kvmsetup{<options>}</code>
	This function sets key-value pairs. Please note that you may need a prefix like <code>bundle</code> in front of the key. Currently the range of supported keys is very limited, but you will learn about them in the description of <code>\bundle</code> .
	<code>\begin{kvmatrix}[<key-value pairs>]</code> <code> <environment content></code> <code>\end{kvmatrix}</code>
<code>kvmatrix</code>	This environment is a semantic interface to <code>tikzpicutre</code> which should have a <code>kvmatrix</code> or <code>\kvlist</code> as first child element. Basically this should surround every Karnaugh map you typeset.
	<code>\begin{kvmatrix}{<a,b,c[...]>}</code> <code> <environment content></code> <code>\end{kvmatrix}</code>
<code>kvmatrix</code>	This environment is one of the two input modes. It provides a structured way of inputting rows and columns, similar to the <code>tabular</code> environment. You should prefer this over <code>\kvlist</code> . The environment itself takes one argument: a comma-separated list of variable names. Please note that they are typeset in math-mode, so you should not use \$ signs.
<code>\kvlist</code>	<code>\kvlist{<width>}{<height>}{<1,o[...]>}{<a,b[...]>}</code>
	This function provides the alternative input mode. You specify width and height of the matrix and then input your elements row-wise. The last argument consists of a comma-separated list of variable names. They are typeset in math-mode.

2.2 Drawing Bundles (implicants)

Another feature of this package is to draw bundles, at least that is how I prefer to call rectangles visualizing adjacent ones or zeros (also called implicants). Please note that currently this package does not compute the bundles for you.

`\bundle` `[\langle key-value pairs \rangle]{\langle x_1 \rangle}{\langle y_1 \rangle}{\langle x_2 \rangle}{\langle y_2 \rangle}`

This function draws a rectangle (bundle) around the area specified by the two corners. The option `invert` opens the bundle outwards and `overlapmargins` lets you specify a length which describes how far the edges will be drawn into the margin (both options are useful for corners). With `color` you may change the color of the border and `reducespace` allows you to specify whether you want the package to be narrower or wider.

When using `invert`, the package tries to determine where to invert automatically. Sometimes it fails with the guesswork. You can manually disable the horizontal inversion part with `hinvert=false` (idem for the vertical inversion part with `vinvert=false`).

Warning: This package is unable to draw a bundle including all four corners this way. If you need this specific edge case, please use TikZ to draw it yourself (see the last example in [section 3](#)).

2.3 Styling the nodes

This package defines two TikZ styles to allow easy customizations: `kvnode` and `kvbundle`. The former is applied to every node within the output matrix, i.e. the zeros and ones. It styles a node, so all options applicable to a `\node` are available to you.

The latter describes the styling of the bundles' paths. It is applied to a `\draw` command, so you have to use options available to paths.

The whole Karnaugh map is a TikZ grid with many TikZ nodes. In the case you need to address a single node you may refer to it by its position, i.e. the node's name is `xy` (zero-based), e.g. `00` for the node in the upper left corner.

3 Examples

```

1 \kvlst{2}{4}{0,1,0,0,0,0,1}{a,b,c}
2 \hfill
3 \begin{kvmatrix}{a,b,c}
4   0 & 1 & 1 & 0 \\
5   0 & 1 & 0 & 1 \\
6 \end{kvmatrix}

```

	<i>a</i>		
<i>bc</i>		0	1
00		0	1
01		0	0
11		0	0
10		0	1

		<i>ab</i>				
<i>c</i>			00	01	11	10
0		0	1	1	0	
1		0	1	0	1	

```

1 \kvlst{4}{4}{0,1,1,0,0,1,0,1,0,1,1,1,0,0,1,1}{a,b,c,d}

```

	<i>ab</i>				
<i>cd</i>		00	01	11	10
00		0	1	1	0
01		0	1	0	1
11		0	1	1	1
10		0	0	1	1

```

1 \begin{kvmmap}
2   \kvlst{4}{4}{0,1,1,0,0,1,0,1,0,1,1,1,0,0,1,1}{a,b,c,
   ↪ ,d,e,f}
3   \bundle[color=red]{1}{1}{1}{2}
4   \bundle{3}{2}{2}{3}
5   \bundle[color=blue, reducespace=3pt]{3}{2}{3}{1}
6 \end{kvmmap}

```

		<i>ab</i>			
		00	01	11	10
<i>cd</i>	00	0	1	1	0
	01	0	1	0	1
	11	0	1	1	1
	10	0	0	1	1

```

1 \begin{kvmatrix}
2   \begin{kvmatrix}{a,b,c,d}
3     0 & 1 & 1 & 0 \\
4     1 & 0 & 0 & 1 \\
5     0 & 0 & 0 & 1 \\
6     0 & 1 & 1 & 1 \\
7   \end{kvmatrix}
8   \bundle{3}{3}{2}{3}
9   \bundle[color=blue]{3}{2}{3}{1}
10  \bundle[invert=true, reducespace=2pt, overlapmargins=6,
    ↪ pt]{1}{0}{2}{3}
11  \bundle[invert=true, reducespace=2pt]{0}{1}{3}{1}
12 \end{kvmatrix}

```

$cd \backslash ab$	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	0	0	0	1
10	0	1	1	1

```

1 \begin{kvmatrix}
2   \begin{kvmatrix}{a,b,c,d,e,f}
3     0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
4     1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
5     1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
6     1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
7     0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
8     0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
9     0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
10    1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
11  \end{kvmatrix}
12  \draw[fill=white, opacity=.6] (00.center) rectangle
13  ↪ (22.center);
14  \draw[white, line width=.4cm] (07) -- (70);
15  \draw[fill=white, opacity=.6] (77.center) rectangle
16  ↪ (55.center);
17 \end{kvmatrix}

```

<i>def</i> \ <i>abc</i>	000	001	011	010	110	111	101	100
000	0	1	1	0	0	1	0	1
001	1	1	1	0	0	1		0
011	1	0	0	1	0		0	1
010	1	0	0	1		0	1	1
110	0	1	0		0	1	1	1
111	0	1		0	1	1	0	0
101	0		0	1	1	1	0	0
100	1	1	0	1	0	1	0	1

4 Implementation

```
1 \RequirePackage{amsmath}
2 \RequirePackage{xparse}
3 \RequirePackage{tikz}
```

```
\seq_set_split:Nnx Define variants for better expansion.
  \int_mod:VV      4 \cs_generate_variant:Nn \seq_set_split:Nnn { Nnx }
\int_div_truncate:VV 5 \cs_generate_variant:Nn \int_mod:nn      { VV  }
6 \cs_generate_variant:Nn \int_div_truncate:nn { VV  }
7
```

*(End definition for `\seq_set_split:Nnx`, `\int_mod:VV`, and `\int_div_truncate:VV`.
These functions are documented on page ??.)*

4.1 Generating the Gray code

This part of the code is primarily based on Marcel Krüger's StackExchange post (see <https://tex.stackexchange.com/questions/418853/latex3-pad-something-unexpandable>).

```
\@@_graycode_xor_bits:nn This function will apply xor to two bits.
#1 : bit (0 or 1)
#2 : bit (0 or 1)
8 \cs_new:Npn \@@_graycode_xor_bits:nn #1#2
9 {
10   \int_compare:nTF { #1 = #2 }
11     { 0 } { 1 }
12 }
```

(End definition for `\@@_graycode_xor_bits:nn`.)

```
\@@_graycode_xor:w This macro executes a bitwise xor on two expanded bit sequences. It
\@@_graycode_xor:nn is defined recursively, so that on every run the first two bits from the
\@@_graycode_xor:xx bit sequence are split.
#1 : first bit sequence
#2 : second bit sequence
```

```

13 \cs_new:Npn \@@_graycode_xor:w #1#2\q_stop#3#4\q_stop
14 {
15   \@@_graycode_xor_bits:nn { #1 } { #3 }
16   \tl_if_empty:nF { #2 }
17   {
18     \@@_graycode_xor:w #2\q_stop#4\q_stop
19   }
20 }
21 \cs_new:Npn \@@_graycode_xor:nn #1#2
22 {
23   \@@_graycode_xor:w #1\q_stop#2\q_stop
24 }
25 \cs_generate_variant:Nn \@@_graycode_xor:nn { xx }
26

```

(End definition for \@@_graycode_xor:w and \@@_graycode_xor:nn.)

\kvmap_graycode_at:nn Calculate the gray code at a specific digit, see the Wikipedia for details.

#1 : digit

#2 : number of bits

```

27 \cs_new:Npn \kvmap_graycode_at:nn #1#2
28 {
29   \@@_graycode_xor:xx
30   { \tl_tail:f { \int_to_bin:n { #1 - 1 + #2 } } }
31   { \tl_tail:f { \int_to_bin:n { \fp_eval:n { floor((#1-1)/2) + #2 } } } }
32 }
33 \cs_generate_variant:Nn \kvmap_graycode_at:nn { nV }
34

```

(End definition for \kvmap_graycode_at:nn. This function is documented on page ??.)

4.2 Internal code to automate output

\l_@@_matrix_isintikz_bool Are we in a tikzpicture?

```

35 \bool_new:N \l_@@_matrix_isintikz_bool

```

(End definition for \l_@@_matrix_isintikz_bool.)

\l_@@_matrix_height_int Save the dimensions of the matrix (important for bundling).

\l_@@_matrix_width_int

```

36 \int_new:N \l_@@_matrix_height_int
37 \int_new:N \l_@@_matrix_width_int

```

(End definition for `\l_@@_matrix_height_int` and `\l_@@_matrix_width_int`.)

`\@@_outputgraycode:` Output gray code around the grid.

```
38 \cs_new:Nn \@@_outputgraycode:
39 {
```

Iterate through the horizontal part first and add each item as node. Add -1 to the coordinate calculation to convert one-based to zero-based numbers.

```
40   \int_step_inline:nnnn { 1 } { 1 } { \l_@@_matrix_width_int }
41   {
42 \node ~ at ~ (\fp_eval:n { 0.5 + (##1-1) }, .3)
43 { \kvmmap_graycode_at:nV { ##1 } \l_@@_matrix_width_int };
44   }
```

Afterwards tackle the vertical part.

```
45   \int_step_inline:nnnn { 1 } { 1 } { \l_@@_matrix_height_int }
46   {
47 \node[anchor = east] ~ at ~ (0, \fp_eval:n { -0.5 - (##1-1) })
48 { \kvmmap_graycode_at:nV { ##1 } \l_@@_matrix_height_int };
49   }
50 }
```

(End definition for `\@@_outputgraycode:`.)

`\@@_outputmatrix:n` Define a TikZ style for easier customizability.

```
52 \tikzset{kvnode/.style = { minimum ~ width=.75cm, minimum ~ height=.75cm }}
```

This macro fill the grid with values.

#1: list

```
53 \cs_new:Npn \@@_outputmatrix:n #1
54 {
```

We iterate using a for each loop, hence there is no counter and we need to define one.

```
55 \int_zero:N \l_tmpa_int
```

Use a temporary sequence to store the argument. This has to be a split because setting from `clist` would eliminate empty elements.

```
56 \seq_set_split:Nnn \l_tmpa_seq { , } { #1 }
```

Loop over the elements of the list. Every element will be output as node where $x = \text{counter} \bmod \text{width}$ and $y = \left\lfloor \frac{\text{counter}}{\text{height}} \right\rfloor$.

```

57 \seq_map_inline:Nn \l_tmpa_seq
58 {
59 \node[kvnode] ~
60 (\int_mod:VV \l_tmpa_int \l_@@_matrix_width_int
61 \int_div_truncate:VV \l_tmpa_int \l_@@_matrix_width_int ) ~
62 at ~
63 (.5+\int_mod:VV \l_tmpa_int \l_@@_matrix_width_int,
64 -.5-\int_div_truncate:nn \l_tmpa_int \l_@@_matrix_width_int ) ~
65 {\smash[b]{\makebox[opt]{###1}}};
66 \int_incr:N \l_tmpa_int
67 }
68 }
69

```

(End definition for `\l_@@_outputmatrix:n`.)

4.3 Implicant-related code

`\bundle` Draw a bundle with given corners.

```

#1 : key-value pairs
#2 : x coordinate of point 1
#3 : y coordinate of point 1
#4 : x coordinate of point 2
#5 : y coordinate of point 2

70 \keys_define:nn { kvmap/bundle }
71 {

```

`reducespace`: reduce inner sep of the node; negative values mean expansion

```

72 reducespace .dim_set:N = \l_@@_bundle_reducespace_dim,
73 reducespace .initial:n = { opt },

```

`color`: path color of the bundle

```

74 color .tl_set:N = \l_@@_bundle_color_tl,
75 color .initial:n = { black },

```

invert: open bundle instead of closed path

```
76 invert      .bool_set:N    = \l_@@_bundle_invert_bool,  
77 invert      .default:n     = true,  
78 invert      .initial:n     = false,
```

vinvert: perform inversion vertically

```
79 vinvert     .bool_set:N    = \l_@@_bundle_vinvert_bool,  
80 vinvert     .default:n     = true,  
81 vinvert     .initial:n     = true,
```

hinvert: perform inversion horizontally

```
82 hinvert     .bool_set:N    = \l_@@_bundle_hinvert_bool,  
83 hinvert     .default:n     = true,  
84 hinvert     .initial:n     = true,
```

overlapmargins: intrude into margin (when inverted)

```
85 overlapmargins .dim_set:N    = \l_@@_bundle_overlapmargins_dim,  
86 overlapmargins .initial:n     = { opt },  
87 }
```

TikZ-style to easily adapt the bundle design.

```
88 \tikzset{kvbundle/.style = { rounded ~ corners = 5pt }}
```

Auxiliary variables for drawing bundles.

```
\l_@@_bundle_minx_int 89 \int_new:N \l_@@_bundle_minx_int  
\l_@@_bundle_miny_int 90 \int_new:N \l_@@_bundle_miny_int  
\l_@@_bundle_maxx_int 91 \int_new:N \l_@@_bundle_maxx_int  
\l_@@_bundle_maxy_int 92 \int_new:N \l_@@_bundle_maxy_int  
  
93 \NewDocumentCommand { \bundle } { O{} m m m m }  
94 {  
95 \group_begin:
```

Set optional parameters and save the minima and maxima of x- and y-coordinates.

```
96 \keys_set:nn { kvmap/bundle } { #1 }  
97 \int_set:Nn \l_@@_bundle_minx_int { \int_min:nn { #2 } { #4 } }  
98 \int_set:Nn \l_@@_bundle_miny_int { \int_min:nn { #3 } { #5 } }  
99 \int_set:Nn \l_@@_bundle_maxx_int { \int_max:nn { #2 } { #4 } }  
100 \int_set:Nn \l_@@_bundle_maxy_int { \int_max:nn { #3 } { #5 } }
```

Check whether the bundle will be inverted. The current conformance test which is executed later on allows the edge case of inverting at every corner, which is currently unsupported (code-wise).

```
101 \bool_if:NTF \l_@@_bundle_invert_bool
102 {
```

If the bundle will be inverted, the first check is whether it will exceed the height. In that case there will be an opened rectangle on both sides (top and bottom) which is positioned according to the minima and maxima of the coordinates. The reducespace option is realized as shift.

```
103 \bool_if:nT
104 {
105 \int_compare_p:n { \l_@@_matrix_height_int - 1 = \l_@@_bundle_maxy_int }
106 && \int_compare_p:n { 0 = \l_@@_bundle_minx_int }
107                                     && \l_@@_bundle_vinvert_bool
108 }
109 {
110 \draw[draw=\l_@@_bundle_color_tl,kvbundle] ~
111 ([xshift=\l_@@_bundle_reducespace_dim,
112 yshift=\l_@@_bundle_overlapargins_dim]
113 \int_use:N \l_@@_bundle_minx_int
114 \int_use:N \l_@@_bundle_miny_int . north ~ west) --
115 ([xshift=\l_@@_bundle_reducespace_dim,
116 yshift=\l_@@_bundle_reducespace_dim]
117 \int_use:N \l_@@_bundle_minx_int
118 \int_use:N \l_@@_bundle_miny_int . south ~ west) --
119 ([xshift=-\l_@@_bundle_reducespace_dim,
120 yshift=\l_@@_bundle_reducespace_dim]
121 \int_use:N \l_@@_bundle_maxx_int
122 \int_use:N \l_@@_bundle_miny_int . south ~ east) --
123 ([xshift=-\l_@@_bundle_reducespace_dim,
124 yshift=\l_@@_bundle_overlapargins_dim]
125 \int_use:N \l_@@_bundle_maxx_int
126 \int_use:N \l_@@_bundle_miny_int . north ~ east);
127 \draw[draw=\l_@@_bundle_color_tl,kvbundle] ~
128 ([xshift=\l_@@_bundle_reducespace_dim,
129 yshift=-\l_@@_bundle_overlapargins_dim]
130 \int_use:N \l_@@_bundle_minx_int
131 \int_use:N \l_@@_bundle_maxy_int . south ~ west) --
132 ([xshift=\l_@@_bundle_reducespace_dim,
```

```

133 yshift=-\l_@@_bundle_reducespace_dim]
134 \int_use:N \l_@@_bundle_minx_int
135 \int_use:N \l_@@_bundle_maxy_int . north ~ west) --
136 ([xshift=-\l_@@_bundle_reducespace_dim,
137 yshift=-\l_@@_bundle_reducespace_dim]
138 \int_use:N \l_@@_bundle_maxx_int
139 \int_use:N \l_@@_bundle_maxy_int . north ~ east) --
140 ([xshift=-\l_@@_bundle_reducespace_dim,
141 yshift=-\l_@@_bundle_overlapmargins_dim]
142 \int_use:N \l_@@_bundle_maxx_int
143 \int_use:N \l_@@_bundle_maxy_int . south ~ east);
144 }

```

Is it larger than the width? Then turn 90° and work as above.

```

145 \bool_if:nT
146 {
147 \int_compare_p:n { \l_@@_matrix_width_int - 1 = \l_@@_bundle_maxx_int }
148 && \int_compare_p:n { 0 = \l_@@_bundle_minx_int }
149                                     && \l_@@_bundle_hinvert_bool
150 }
151 {
152 \draw[draw=\l_@@_bundle_color_tl,kvbundle] ~
153 ([yshift=-\l_@@_bundle_reducespace_dim,
154 xshift=-\l_@@_bundle_overlapmargins_dim]
155 \int_use:N \l_@@_bundle_minx_int
156 \int_use:N \l_@@_bundle_miny_int . north ~ west) --
157 ([yshift=-\l_@@_bundle_reducespace_dim,
158 xshift=-\l_@@_bundle_reducespace_dim]
159 \int_use:N \l_@@_bundle_minx_int
160 \int_use:N \l_@@_bundle_miny_int . north ~ east) --
161 ([yshift=\l_@@_bundle_reducespace_dim,
162 xshift=-\l_@@_bundle_reducespace_dim]
163 \int_use:N \l_@@_bundle_minx_int
164 \int_use:N \l_@@_bundle_maxy_int . south ~ east) --
165 ([yshift=\l_@@_bundle_reducespace_dim,
166 xshift=-\l_@@_bundle_overlapmargins_dim]
167 \int_use:N \l_@@_bundle_minx_int
168 \int_use:N \l_@@_bundle_maxy_int . south ~ west);
169 \draw[draw=\l_@@_bundle_color_tl,kvbundle] ~
170 ([yshift=-\l_@@_bundle_reducespace_dim,
171 xshift=\l_@@_bundle_overlapmargins_dim]

```

```

172 \int_use:N \l_@@_bundle_maxx_int
173 \int_use:N \l_@@_bundle_minx_int . north ~ east) --
174 ([yshift=-\l_@@_bundle_reducespace_dim,
175 xshift=\l_@@_bundle_reducespace_dim]
176 \int_use:N \l_@@_bundle_maxx_int
177 \int_use:N \l_@@_bundle_minx_int . north ~ west) --
178 ([yshift=\l_@@_bundle_reducespace_dim,
179 xshift=\l_@@_bundle_reducespace_dim]
180 \int_use:N \l_@@_bundle_maxx_int
181 \int_use:N \l_@@_bundle_maxy_int . south ~ west) --
182 ([yshift=\l_@@_bundle_reducespace_dim,
183 xshift=\l_@@_bundle_overlapargins_dim]
184 \int_use:N \l_@@_bundle_maxx_int
185 \int_use:N \l_@@_bundle_maxy_int . south ~ east);
186 }
187 }
188 {

```

If the package will not be inverted, it will be output by using the coordinates provided (min and max).

```

189 \draw[draw=\l_@@_bundle_color_tl, kvbundle] ~
190 ([xshift=\l_@@_bundle_reducespace_dim,
191 yshift=-\l_@@_bundle_reducespace_dim]
192 \int_use:N \l_@@_bundle_minx_int
193 \int_use:N \l_@@_bundle_minx_int . north ~ west) ~
194 rectangle ~
195 ([xshift=-\l_@@_bundle_reducespace_dim,
196 yshift=\l_@@_bundle_reducespace_dim]
197 \int_use:N \l_@@_bundle_maxx_int
198 \int_use:N \l_@@_bundle_maxy_int . south ~ east);
199 }
200 \group_end:
201 }

```

(End definition for \bundle and others. These functions are documented on page 4.)

4.4 User-interface code

```

\kvmap_map:nn Output the matrix (interface level).
\kvmap_map:xn #1 : values

```


#2 : variables (lables), first horizontal part, then vertical

Example: `\kvmap_map:nn{0,1,1,0,0,1,0,1,0,1,1,1,0,0,1,1}{a,b,c,d}`

```
202 \cs_new:Npn \kvmap_map:nn #1#2
203 {
```

Firstly, generate the map (lines).

```
204 \draw ~ (0,0) ~ grid ~
205 (\int_use:N \l_@@_matrix_width_int, -\int_use:N \l_@@_matrix_height_int);
```

After having drawn the grid, construct the nodes from the csv and output the gray code.

```
206 \@@_outputmatrix:n { #1 }
207 \@@_outputgraycode:
208 \draw ~ (0,0) ~ -- ~ (-.7,.7);
```

Now the labels will be output. The first part of the csv will be positioned at the top right, the second part bottom left. The point of separation will be saved.

```
209 \int_set:Nn \l_tmpa_int
210 { \fp_eval:n { floor(ln(\l_@@_matrix_width_int)/ln(2)) } }
211 \tl_clear:N \l_tmpa_tl
212 \int_step_inline:nnnn { 1 } { 1 } { \l_tmpa_int }
213 {
214 \tl_put_right:Nn \l_tmpa_tl { \clist_item:nn { #2 } { ##1 } }
215 }
216 \node[anchor = west] ~ at ~ (-.5, .7) ~ { $\tl_use:N \l_tmpa_tl$ };
```

After working on the horizontal part, work on the left entries. This will also become a node.

```
217 \tl_clear:N \l_tmpa_tl
218 \int_step_inline:nnnn { \l_tmpa_int + 1 } { 1 }
219 { \l_tmpa_int + \fp_eval:n { floor(ln(\l_@@_matrix_height_int)/ln(2)) } }
220 {
221 \tl_put_right:Nn \l_tmpa_tl { \clist_item:nn { #2 } { ##1 } }
222 }
223 \node[anchor = east] ~ at ~ (-.4, .2) ~ { $\tl_use:N \l_tmpa_tl$ };
```

```
224 }
225 \cs_generate_variant:Nn \kvmap_map:nn { xn }
226
```

(End definition for `\kvmap_map:nn`. This function is documented on page ??.)

`kvmap` This is a `tikzpicture`, but for semantic reasons introduced as new environment. Furthermore this will clear the results of the last execution.
#1 : optional: key-value pairs

```

227 \NewDocumentEnvironment { kvmap } { 0{ } }
228 {
229 \group_begin:
230 \keys_set:nn { kvmap } { #1 }
231 \int_gzero:N \l_@@_matrix_height_int
232 \int_gzero:N \l_@@_matrix_width_int
233 \begin{tikzpicture}
234 }
235 {
236 \end{tikzpicture}
237 \group_end:
238 }

```

`\kvlist` User wrapper around `\kvmap_map:nn`. It will insert a `tikzpicture` if not already present.

```

239 \NewDocumentCommand { \kvlist } { m m m m }
240 {
241 \tikzifinpicture
242 { \bool_set_true:N \l_@@_matrix_isintikz_bool }
243 { \bool_set_false:N \l_@@_matrix_isintikz_bool }
244 \bool_if:NF \l_@@_matrix_isintikz_bool
245 { \begin{tikzpicture} }
246 \int_gset:Nn \l_@@_matrix_width_int { #1 }
247 \int_gset:Nn \l_@@_matrix_height_int { #2 }
248 \kvmap_map:nn { #3 } { #4 }
249 \bool_if:NF \l_@@_matrix_isintikz_bool
250 { \end{tikzpicture} }
251 }
252

```

(End definition for `\kvlist`. This function is documented on page 3.)

`kvmatrix` This environment enables a tabular-like input syntax.
#1 : labels (variables)

`\l_@@_tmp_seq` Temporary variable to split the matrix.

```

253 \seq_new:N \l_@@_tmp_seq

```

(End definition for `\l_@_tmp_seq`.)

```
254 \NewDocumentEnvironment { kvmatrix } { m +b }  
255 {
```

Split the environments body at `\\` and remove empty lines. Now the height of the map is just the count of the sequence. Split the first element at `&` and use the count of that as width.

```
256 \seq_set_split:Nnn \l_tmpa_seq { \\ } { #2 }  
257 \seq_remove_all:Nn \l_tmpa_seq { }  
258 \seq_set_split:Nnx \l_tmpb_seq { & } { \seq_item:Nn \l_tmpa_seq { 1 } }  
259 \int_gset:Nn \l_@_matrix_width_int { \seq_count:N \l_tmpb_seq }  
260 \int_gset:Nn \l_@_matrix_height_int { \seq_count:N \l_tmpa_seq }
```

Clean up the lists and convert the input into something `\kvmap_map:nn` may process. Maybe this could be optimised in terms of performance (TikZ `matrix?`).

```
261 \seq_clear:N \l_@_tmp_seq  
262 \seq_map_inline:Nn \l_tmpa_seq  
263 {  
264 \seq_clear:N \l_tmpb_seq  
265 \seq_set_split:Nnn \l_tmpb_seq { & } { ##1 }  
266 \seq_concat:NNN \l_@_tmp_seq \l_@_tmp_seq \l_tmpb_seq  
267 }
```

Output the map by inserting commas between the elements of the temporary sequence.

```
268 \tikzifinpicture  
269 { \bool_set_true:N \l_@_matrix_isintikz_bool }  
270 { \bool_set_false:N \l_@_matrix_isintikz_bool }  
271 \bool_if:NF \l_@_matrix_isintikz_bool  
272 { \begin{tikzpicture} }  
273 \kvmap_map:xn  
274 { \seq_use:Nnnn \l_@_tmp_seq { , } { , } { , } }  
275 { #1 }  
276 \bool_if:NF \l_@_matrix_isintikz_bool  
277 { \end{tikzpicture} }  
278 } { }  
279
```

`\kvmapsetup` Set options (key-value pairs). Example: `\kvmapsetup{bundle/color=red}`

#1 : key-value pairs

```
280 \NewDocumentCommand { \kvmapsetup } { m }  
281 {  
282 \keys_set:nn { kvmap } { #1 }  
283 }
```

(End definition for \kvmapsetup. This function is documented on page 3.)

Change History

vo.1.0		vo.3.2	
General: Stable release	1	$\@@_outputmatrix:n$: Allow	
vo.1.1		empty elements	11
\bundle : Correct dimensions	12	$kvmatrix$: Count empty	
vo.2.0		columns	18
General: Rework		vo.3.3	
documentation and		\bundle : Introduce $hinvert$	
syntax recommendations	1	and $vinvert$	12
vo.2.1		vo.3.4	
\bundle : Fix inversion	12	$\@@_outputmatrix:n$: Fix	
vo.3.0		node size	11
General: Improve		vo.3.5	
documentation	1	General: Add $amsmath$	
vo.3.1		dependency to have	
General: Fix INS file	1	$smash$ option b available	9