

# tabular2 宏包示例文档

Ms\_yam

2026 年 3 月 18 日

## 摘要

本文档是 tabular2 宏包文档的扩展文档，用于展示 tabular2 的一些基本功能。同时，本文档也是 tabular2 的功能测试文档。文档本身只作功能介绍及效果展示，并未直接展示生成表格的相关代码，需要用户自行查看源代码<sup>1</sup>。查看源代码之前，应当先阅读 tabular2 宏包文档的用户部分。

## 目录

<b>1</b>	<b>行列名设置</b>	<b>2</b>
1.1	行与列的名称	2
1.2	表头与行列名	3
1.3	示例	3
<b>2</b>	<b>数据录入环境</b>	<b>3</b>
2.1	标准输入	4
2.1.1	默认效果	4
2.1.2	特殊字符	4
2.2	CSV 输入	4
2.2.1	默认效果	5
2.2.2	特殊字符【待处理】	5
2.3	JSON 输入	5
2.3.1	默认效果	5
2.3.2	特殊字符【待处理】	5
2.4	输入位置	6
2.5	无表头输入	6
2.6	文件导入【待实现】	7

---

<sup>1</sup>现在编辑器和 PDF 阅读器均支持反向查看，要找对应的源代码几乎无难度。

1 行列名设置	2
3 行列与单元格输入	7
3.1 按行输入	7
3.2 按列输入	8
3.3 按单元格输入	8
4 保存与加载	8
4.1 保存表格	8
4.2 加载表格	9
5 格式设置	9
5.1 宽度与高度	9
5.2 对齐方式	10
6 合并单元格	10
7 渲染输出	10
7.1 直接打印	11
7.2 三线表	11
7.3 栅格模式	11
7.4 全边框模式	12

# 1 行列名设置

## 1.1 行与列的名称

为了便于定位，本宏包的中表格的行与列均可以设置名称。任意一行或一列可以有多个名称，但一个行名或列名则只能映射到唯一的一行或一列；然而这种映射却是动态的，如某个行名可以在某个位置指向第 1 行，而在另外一个位置指向第 2 行。

命令 `\rowname` 与 `\colname` 专用用来指定行名与列名，它们会自动扩展表格的大小（行数或列数）。如果您习惯了 Excel 的 A1 样式，则可以通过 `\excelcolname` 命令来初始化列名<sup>2</sup>。

虽然我们可以使用纯数字<sup>3</sup>来设置行名与列名，但这样的名称不能用于定位<sup>4</sup>。因此，除非要设置表头，否则建议使用字母来设置行名与列名。

<sup>2</sup>它不会影响表格的大小。

<sup>3</sup>在内部实现中，所有名称都按凭据表处理（包括纯数字）。

<sup>4</sup>在定位时，会优先尝试将坐标解释为数字坐标，失败后才会按名称查找映射表。

## 1.2 表头与行列名

表格有行表头与列表头；默认以行名为行表头；以列名为列表头。如果某一个行设置了多个行名，则以最后一个行名为准；列表头类似。

当显示指定行表头或列表头时，则不受行列名的影响。

默认情况下，只输出列表头，不输出行表头。表头是否输出则可通过相关设置来调整。[To do \(1\)](#)

## 1.3 示例

由于宏包内部使用两种完全不同的输出方式，为保证测试的覆盖程度，本文档中的所有输出示例，默认均以全边框形式及三线表形式两种方式输出。

以下是第一个示例，用于观察行名与列名设置效果：（包含行列表头）

title	5	B	AA	BB	CC
2					
A					data

title	5	B	AA	BB	CC
2					
A					data

说明：

- 此时行名的映射为：<2> = 1（无效），<A> = 2；如果这时输入行坐标 2，它会识别为第 2 行而非第 1 行；
- 对于列名同理，虽然列名 <5> = 1，但因其不生效，所以最终修改的是第 5 列；
- 行列名中所有字符均会被当作字符串处理，如 `\footnote` 等；
- 对于表头来说，以上设置却是有生效的；<sup>5</sup>

## 2 数据录入环境

`data` 环境是主要的数据输入环境。目前该环境支持标准输入、CSV 输入及 JSON 输入三种输入方式。同时，CSV 与 JSON 还可以考虑支持文件输入。[To do \(2\)](#)

<sup>5</sup>脚注等命令，需要通过显示设置行列表头才生效。

## 2.1 标准输入

标准输入是本宏包设计的主要输入方式。该输入方式采用硬解码，处理速度有较明显优势，且直接支持行列表头输入，应当作为常规表格的首选输入方式。

### 2.1.1 默认效果

以下是全部使用默认值（逗号分隔，\\ 换行）的标准输入的效果：

Name	Sex	Age <sup>6</sup>	Name	Sex	Age <sup>7</sup>
John	man	18	John	man	18
Leon		20	Leon		20
Lily	woman	21	Lily	woman	21

其中：首行为表头（兼列名），中间的行会保留，尾随的空行不会。如果某行在某列之后无需设置，则允许省略该行的后续分隔符。

### 2.1.2 特殊字符

标准输入不对特殊字符进行任何预处理，所有输入均为凭据表，直接由  $\text{T}_{\text{E}}\text{X}$  的默认方式处理。但由于“\\”用于分隔表格行，因此单元格换行需要使用“\newline”。如果单元格本身带逗号，则可通过指定其它分隔符（选项“sep”）来规避。

AA	BB	CC	AA	BB	CC
	a			a	
A{[#\$	b	”_	A{[#\$	b	”_
B}}%&	c		B}}%&	c	
d	,\		d	,\	

## 2.2 CSV 输入

CSV 输入与标准输入非常相似，但是它会处理特殊字符<sup>8</sup>；代价则是处理速度会更慢。在没有特殊字符的情况下，应当尽量使用标准输入。

<sup>6</sup>这是一条演示用的脚注。

<sup>7</sup>这是一条演示用的脚注。

<sup>8</sup>目的是尽量匹配 CSV 的输入语法。

2.2.1 默认效果

通过指定 “format=csv” 选项（或直接使用 “csv”），可让本环境按 CSV 格式处理输入。

AA	BB	CC\footnote {}
A	ab	1
B	cd	2
		3 <sup>9</sup>

AA	BB	CC\footnote {}
A	ab	1
B	cd	2
		3 <sup>10</sup>

虽然本宏包也支持指定分隔符，但是仍建议遵守约定，使用默认值 “,”。

2.2.2 特殊字符【待处理】

To do (3)

2.3 JSON 输入

JSON 的输入方式与标准输入迥异，不需要使用 “\\” 来换行。其属性名为列名，属性值为表体。

2.3.1 默认效果

通过指定 “format” 选项为 “json”，可让本环境处理 JSON 格式。

A	B	C	D\footnote {}
a	true	1	
bc		2	
d		3	不支持 TeX 命令
e	false	4	

A	B	C	D\footnote {}
a	true	1	
bc		2	
d		3	不支持 TeX 命令
e	false	4	

2.3.2 特殊字符【待处理】

To do (4)

<sup>9</sup>暂时只有内容才支持脚注。

<sup>10</sup>暂时只有内容才支持脚注。

## 2.4 输入位置

可以通过“loc”选项来指定输入的起点位置<sup>11</sup>；如果未指定，则默认为“{1, 1}”。

对于标准输入（含 CSV 格式），输入过程中，会根据输入列名依次更新列名映射。如本例中的“Name”，它原本映射到第 1 列，但输入后映射到第 3 列<sup>12</sup>。

		Name	Sex	Age		Name	Sex	Age
Old					Old			
		John	man	18		John	man	18
		Leon		20		Leon		20
		Lily	woman	21		Lily	woman	21

但对于 JSON 格式，则是完全不一样的逻辑：如果列名已经存在，则直接填充数据；如果列名不存在，则在指定位置及之后依次添加新列名。<sup>13</sup>

Sex	Name	Age	Sex	Name	Age
man	John		man	John	
man	John	18	man	John	18
	Leon	20		Leon	20
woman	Lily	21	woman	Lily	21

## 2.5 无表头输入

在标准输入与 CSV 输入中，默认第一行为表头兼列名；但这可能通过将“header”选项设为“false”来将第一行当作数据行处理。

A	B	C	A	B	C
Name	Sex	Age	Name	Sex	Age
John	man	18	John	man	18
Leon		20	Leon		20
Lily	woman	21	Lily	woman	21

<sup>11</sup>这可能会造成位置之前的列没有列名，在这种情况下其表头空。

<sup>12</sup>虽然映射更新了，但它不会影响之前的录入，如单元格“(1,1)”。

<sup>13</sup>JSON 中的属性名可以是无序的，所以列名顺序设计上是独立的；而 CSV 有序，因此会自动更新表头。

如果同时还需要将表格设置为无表头，则需要在 `xtable` 层级中设置<sup>14</sup>。

Name	Sex	Age
John	man	18
Leon		20
Lily	woman	21

Name	Sex	Age
John	man	18
Leon		20
Lily	woman	21

如果即想把第一行当作列名又想不输出表头，则需要组合设置该选项。

John	man	18
Leon		20
Lily	woman	21

John	man	18
Leon		20
Lily	woman	21

对于 JSON 格式，由于属性名始终为列名，因此在 `data` 层级不支持这个选项。但在 `xtable` 层级中，仍然支持。

John	man	18
Leon		20
Lily	woman	21

John	man	18
Leon		20
Lily	woman	21

2.6 文件导入【待实现】

`\file` 是的功能与 `data` 的功能几乎一致，除了其是以文件作为输入及换行有特殊处理。<sup>To do (5)</sup>

3 行列与单元格输入

本宏包支持按行、按列及按单元格输入，且这三种方式可以组合使用。输入过程中如果数据超出现有行列数，则会自动添加行列。  
此部分是本宏包与传统表格之间的最大差异点之一。

3.1 按行输入

按行输入（命令 `\row`）用于横向添加数据，可以指定起点位置，默认为新行第一列。以下是按行输入的示例：

<sup>14</sup>此时，`data` 层级默认会继承这个设置，除非手动指定。

Name	Sex	Age	
John	man	18	
Lily		20	
	woman	18	newcol
Jacob	man	21	

Name	Sex	Age	
John	man	18	
Lily		20	
	woman	18	newcol
Jacob	man	21	

### 3.2 按列输入

按行输入（命令 `\col`）用于纵向添加数据，可以指定起点位置，默认为新列第一行。以下是按列输入的示例：

Name	Sex	Age	
A	man	18	1
B	woman		2
	man		

Name	Sex	Age	
A	man	18	1
B	woman		2
	man		

### 3.3 按单元格输入

按单元格输入（命令 `\cell`）用于向指定位置添加数据。以下是按单元格输入的示例：

Name	Sex	Age	newcol
John	man	18	
			newdata

Name	Sex	Age	newcol
John	man	18	
			newdata

## 4 保存与加载

为了提高数据复用率，本宏包提供了表格保存与加载功能。

### 4.1 保存表格

以下是一个基础表，使用保存命令 `\savetable`，然后再修改了部分内容。（保存数据没有直接体现）



Name	Sex	Age
John	man	18
Unsaved		20
Lily	woman	21

Name	Sex	Age
John	man	18
Unsaved		20
Lily	woman	21

请注意，在保存命令之后做的修改不会影响已经保存的表格。

## 4.2 加载表格

以下是复用的表格数据：

Name	Sex	Age
John	man	18
Leon	man	20
Lily	woman	21

Name	Sex	Age
John	man	18
Leon	man	20
Lily	woman	21

注，加载表格会覆盖之前的设置，因此通常把它放置到第一行。

# 5 格式设置

## 5.1 宽度与高度

表格会依次计算各列所需的宽度，然后再计算各行所需的高度<sup>15</sup>，以包容所有单元格。

AA	BB	CC	EE
abcde	12	dd	abccba
	19		
ab	20	xy	abc

AA	BB	CC	EE
abcde	12	dd	abccba
	19		
ab	20	xy	abc

但可以通过 `\rowheight` 与 `\colwidth` 来指定额外的规则或固定尺寸。

AA	BB	CC	EE
abcde	12	dd	abccba
	19		
ab	20	xy	abc

AA	BB	CC	EE
abcde	12	dd	abccba
	19		
ab	20	xy	abc

注：如果宽度设置过小，会导致数据超出单元格。**To do (6)**

<sup>15</sup>这里所说的高度是行的最顶端面到底端的距离，实现中包括 L<sup>A</sup>T<sub>E</sub>X 中的高度与深度两个值。

5.2 对齐方式

表格默认是左右居中,上下底对齐的;也可通用 `\rowalign` 与 `\colalign` 来指定对齐方式。

AA	BB	CC	EE
abcde	12	dd	abccba
ab	19 20	xy	abc

AA	BB	CC	EE
abcde	12	dd	abccba
ab	19 20	xy	abc

6 合并单元格

`TEX` 应当避免使用合并单元格,但为了满足某些需求,本宏包提供了合并单元格的功能。

Name	Sex	Age	remark
John	man	18	test
John	man		test
John	18 <sup>16</sup>		test
John			test
John	man	18	test

Name	Sex	Age	remark
John	man	18	test
John		man	test
John		18 <sup>17</sup>	test
John			test
John	man	18	test

合并单元格的齐还需要进一步研究。 `To do` (7)

7 渲染输出

目前支持四种样式的输出:

- 直接按行打印内容 (纯盒子);
- 三线表 (如上述示例中右边的效果);;
- 栅格模式 (不支持合并单元格);
- 全边框模式 (如上述示例中左边的效果);

<sup>16</sup>这是一个合并单元格。  
<sup>17</sup>这是一个合并单元格。

## 7.1 直接打印

可以使用 `\printtable` 直接打印表格。每行数据使用一个水平盒子输出，且不绘制任何边框。因此它可以自动换页，其效果如下：

Name	Sex	Age	remark
John	man	18	test
John	man		test
John		18 <sup>18</sup>	test
John			test
John	man	18	test

## 7.2 三线表

由于三线表在  $\text{T}_\text{E}_\text{X}$  中比较受欢迎，因此本宏包添加了支持。除了传统的三条线外，本宏包还支持添加额外的水平分隔线，效果如下：

Name	Sex	Age	remark
John	man	18	test
John		man	test
John		18 <sup>19</sup>	test
John			test
John	man	18	test

## 7.3 栅格模式

本渲染模式下，允许用户指定要绘制哪些水平线与垂直线，但它只支持整条线。其效果如下：

Name	Sex	Age	remark
John	man	18	test
John		man	test
John		18 <sup>20</sup>	test
John			test
John	man	18	test

<sup>18</sup> 这是一个合并单元格。

<sup>19</sup> 这是一个合并单元格。

<sup>20</sup> 这是一个合并单元格。

### 7.4 全边框模式

本模板会输出所有边框，包括合并单元格的边框。示例在前面已经讲了，这里就不举例了。

### To do...

- ☐ 1 (p. 3): 更新如何设置表头是否输出;
- ☐ 2 (p. 3): 解决文件输入;
- ☐ 3 (p. 5): 处理 CSV 中的特殊字符
- ☐ 4 (p. 5): 处理 JSON 中的特殊字符
- ☐ 5 (p. 7): 实现文件导入功能
- ☐ 6 (p. 9): 如果盒子高度过小, 会怎么样。
- ☐ 7 (p. 10): 优化合并单元格的对齐。