rowmantic: Tables row by row

A Typst package for editing tables row-by-row.

The idea is a row-oriented way to input tables, with just a little less syntactical overhead than the usual table function in Typst.

The rowtable function works like the usual table function but takes one markup block ([...]) per row, and the markup is split internally on a delimiter which is δ by default.

```
Input: [A \& B \& C]
Table cells (effectively): ..([A], [B], [C])
```

For improved table ergonomics, the table sizes the number of columns by the longest row. All rows are effectively completed so that they are of full length. This creates a better the editing experience, as rows can be filled out gradually.

Examples

Document Result

```
goá iáu-boē
                                   koat-tēng
                                                              tang-sî
                                                                                   boeh
                                                                                                   tńg-khì
            iau<sup>1</sup>-boe<sup>3</sup> koat<sup>2</sup>-teng<sup>3</sup> tang<sup>7</sup>-si<sup>5</sup>
                                                                                                   tna<sup>1</sup>-khi<sup>3</sup>
goa<sup>1</sup>
                                                                                   boeh<sup>2</sup>
                                                                                   boeh<sup>4</sup>
goa<sup>2</sup> iau<sup>2</sup>-boe<sup>7</sup>
                                   koat<sup>4</sup>-teng<sup>7</sup> tang<sup>1</sup>-si<sup>5</sup>
                                                                                                    tng<sup>2</sup>-khi<sup>3</sup>
             not-yet
                                    decide
                                                              when
                                                                                                    return.
                                                                                   want
```

Input

```
#{
  show regex("\d"): super.with(size: 0.8em, typographic: false)
  show table.cell: it => { set text(size: 0.9em) if it.y >= 1; it }
  show table.cell.where(y: 0): emph
  rowtable(
    separator: ",",
                     // configurable separator
    stroke: none,
                     // pass through table arguments, hlines, cells et.c.
    inset: (x: 0em),
    column-gutter: 0.9em,
    // rows are filled to be equal length after collecting cells
                                     tang-sî,
    [goá,
            iáu-boē,
                        koat-tēng,
                                                  boeh,
                                                          tńg-khì
                                                                    ],
    [goa1,
                       koat2-teng3, tang7-si5,
                                                         tng1-khi3 ],
           iau1-boe3,
                                                  boeh2,
    [goa2, iau2-boe7,
                       koat4-teng7, tang1-si5,
                                                  boeh4, tng2-khi3],
            not-yet,
                        decide,
                                      when.
                                                  want,
                                                          return.
    table.hline(),
    // cell that fills remainder of row
    expandcell["I have not yet decided when I shall return."],
  )
}
```

Example from Wikipedia²

[&]quot;I have not yet decided when I shall return."

¹But shallowly - not looking into styled or nested content

²https://en.wikipedia.org/wiki/Interlinear_gloss

Document Result

Term	Explanation	Assumptions
X	Explanatory variables	Non-random
Y	$Y_1,,Y_n$ observations	Pairwise independent
β	Model parameters	

```
#{
 set table(stroke: none, inset: 0.8em)
 set table.hline(stroke: 0.5pt)
 show table.cell.where(y: 0): strong
 show table.cell.where(x: 0): x => math.bold(math.upright(x))
 rowtable(
   table.hline(),
   table.header([Term & Explanation
                                             & Assumptions ]),
   table.hline(),
   [$X$
             & Explanatory variables
                                             & Non-random ],
             & $Y_1, ..., Y_n$ observations & *Pairwise independent*],
   [$beta$ & Model parameters
   table.hline(),
}
```

Trying some more difficult examples

Document Result

Literal &	Strong	X -Y		
Equation $\pi = 3.1415$	$\int_{\Omega}d\omega$	X&Y		
π^1	π^2	π^3		
• A • B	1. A 2. B	A a B b		
Figure 1: Top	See Figure 1 & Figure 2	B Figure 2: Bot		
Nested rowtable	Nested table			
АВ	АВ	table.cell(rowspan: 2)		
Cell with colspan=2				
Expandcell				
N/A	N/A	N/A		

```
#rowtable(
  align: horizon,
  stroke: 0.1pt,
  row-filler: [N/A],
 [Literal \& & *Strong* & *X*--_Y_ ],

[Equation \ $pi = 3.1415...$ & $ integral_Omega d omega $ & $X \& Y$],
  $ pi^1 & pi^2 & pi^3 $,
    – A
– B
    ծ
   + A
   + B
    / A: a
    / B: b
  ],
    #{
      set figure.caption(position: top)
      [#figure(rect[A], caption: "Top")<fig1>]
    See @fig1 \& @fig2
    #figure(rect[B], caption: "Bot")<fig2>
  ],
  {
    [Nested rowtable \ ]
    rowtable([A & B])
    [8]
    [Nested table \ ]
    table(columns: 2, [A], [B])
    table.cell(stroke: 1pt + red, rowspan: 2)[`table.cell(` \ `rowspan: 2)`]
  [#table.cell(fill: yellow.lighten(90%), colspan: 2)[Cell with colspan=2]],
  [#expandcell(fill: yellow.lighten(90%))[Expandcell] & #expandcell[#none]],
  table.footer([]),
```

Double semicolon separator

Document Result

First	This is a literal ;; and ; and , and $\&$
Second; Third	Equation $\pi = 3.1415$

Input

Combine with pillar (or other table function)

Use the table argument to let rowtable pass its result to a different table function rather than the standard one, for example pillar.table (shown below) or zero.ztable.

Document Result

Isotope	Z	N	Half-life	Mass (Da)	Abundance (%)
¹⁰⁷ Ag	47	60	Stable	106.9050915(26)	51.839 ± 0.008
¹⁰⁹ Ag	47	62	Stable	108.904 755 8(14)	48.161 ± 0.008

```
#import "@preview/pillar:0.3.2"
#set text(font: "Libertinus Serif")
#show table.cell.where(y: 0): strong
#rowtable(
 separator: ",",
  table: pillar.table,
 cols: "rCCCCC",
format: (auto, ) * 4 + ((uncertainty-mode: "compact"), auto),
  column-gutter: (0.5em, 0pt),
  stroke: (x, y) \Rightarrow if y == 0 \{ (bottom: 0.75pt) \},
  table.header(
  [Isotope,
                 Z, N, Half-life, Mass (Da),
                                                             Abundance (%) ]),
 [#super[107]Ag, 47, 60, Stable, [#super[109]Ag, 47, 62, Stable,
                                         106.9050915(26), 51.839(8)],
                                         108.9047558(14), 48.161(8)],
```

Equations

Equations as rows are are split on δ by default, but it is configurable. Note that δ can be escaped in equations, but other separator symbols are not as easy to escape³.

Document Result

Α	В	С	D	E
1	\boldsymbol{x}	x^2	\sum_{i}^{n}	inline equation row
1	x	x^2	\sum_{i}^{n}	block equation row
1	x	x^2	\sum_{i}^{n}	markup with embedded equations row

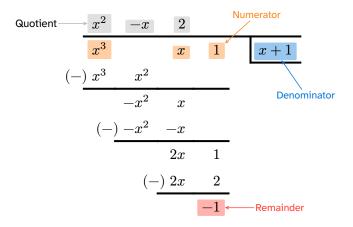
Input

```
#rowtable(
  align: center + horizon,
  stroke: (x, y) => (bottom: int(y == 0) * 1pt),
  separator: ",", // regular sep
  separator-eq: $6$, // equation sep
  [A, B, C, D, E],
  $1 & x & x^2 & sum_i^n & #[inline equation row]$,
  $1 & x & x^2 & sum_i^n & #[block equation row] $,
  [$1$, $x$, $x^2$, $ sum_i^n $, markup with embedded \ equations row],
)
```

Long Division

This example shows one way to set up long division. In this case it's polynomial division, computing the result $x^3+x+1=(x^2-x+2)(x+1)-1$. For this example, the colorful annotations add the most of the complexity. rowtable contributes to the example by splitting equations on the separator and filling rows to be equal length.

Document Result



```
#import "@preview/mannot:0.3.0": annot, markhl
```

 $^{^3}$ The escape for & is just &, for other separators like for example the comma a box or "," is used to escape them.

```
/// Set up strokes and gutter for long division table
#let longdiv(..args, table: std.table) = {
 let cols = args.at("columns")
  let st = std.stroke(args.at("stroke", default: black))
  let stroke = (x, y) \Rightarrow (
    // Add left stroke to the last column
    left: if x == cols - 1 and y == 1 \{ st \},
    bottom: if (
      // Add top and bottom stroke to denominator cell
      y == 1 and x == cols - 1
      // Add bottom stroke every two rows (calc.even check),
      // but for one less column each time
      or y \ge 1 and x < cols - 1 and calc.even(y) and x + 1 \ge y / 2
    ) {
      st
    }
  table(..args,
    column-gutter: (0pt,) * (cols - 2) + (1.5em,),
    stroke: stroke,
    std.table.hline(y: 1, stroke: st))
}
// Set up marking functions
#let markhl = markhl.with(outset: (top: 0.15em, rest: 0.30em), radius: 1pt)
#let um = math.class("unary", math.minus) // unary minus
#let mkq = markhl.with(color: luma(70%))
#let mkn = markhl.with()
#let mkdenom = markhl.with(color: blue, tag: <denom>)
#let mkrem = markhl.with(color: red, tag: <rem>)
#let leftset(x) = box(place(dx: -0.3em, right + bottom, $#x$))
#let rm = math.class("unary", leftset($(-)$)) // row minus
#show: block.with(breakable: false)
#rowtable(
  align: right,
  table: longdiv.with(stroke: 1.5pt),
  inset: 0.55em,
  mkq(x^2, tag: \#<ans>) & mkq(um x) & mkq(2)
                                               გ$.
  $mkn(x^3) &
                       & mkn(x) & mkn(1, tag: \#<num>) & mkdenom(x + 1)$,
  $rm x^3 & x^2$,
            &-x^2
                       ъ x$,
            \delta rm -x^2 \delta -x^4,
  $
                        & 2x & 1$,
            Ֆ
  $
            ծ
                        8rm 2x & 2$,
  $
                               & mkrem(um 1)$,
#annot(<ans>, pos: left + horizon, dx: -2em, dy: -0em, annot-text-props: (fill:
black, size: 0.75em))[Quotient]
#annot(<denom>, pos: right + bottom, dy: +2em)[Denominator]
#annot(<num>, pos: right + top, dy: -1.5em, dx: 1em)[Numerator]
#annot(<rem>, pos: right + horizon, dy: 0em, dx: 2em)[Remainder]
```

Equations with alignment

Use a different separator than & to use equations with alignment.

Document Result

Input

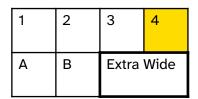
```
#set math.mat(delim: "[")
#rowtable(
    separator: ";",
    stroke: (x, y) => (bottom: int(y == 0) * 1pt),
    [Matrix Form; Equation System Form],
    $
      mat(y_1; y_2) = mat(a_11, a_12; a_21, a_22) mat(x_1; x_2)
    ;
      "(1st)" y_1 &= a_11 x_1 &+ a_12 x_2 \
      "(2nd)" y_2 &= a_21 x_1 &+ a_22 x_2 \
      $
}
```

Table Cells, rowspan and colspan

- colspan: a cell spans multiple columns
- rowspan: a cell spans multiple rows

Table cells can be customized with the usual properties (stroke, fill, et.c.) but also with colspan and rowspan. It's important to include the cells inline inside the rows, i.e like this:

Document Result



Input

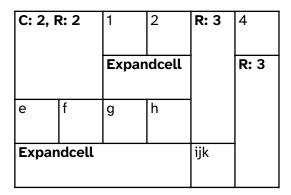
```
#let cell = table.cell
#rowtable(
   separator: ",",
   column-width: 3em,
   rows: 3em,
   [1, 2, 3,  #cell(fill: yellow)[4]],
   [A, B, #cell(colspan: 2, stroke: 2pt)[Extra Wide]],
)
```

where [...] is the markup for the whole row. Then automatic row length computations will continue to work correctly.

The column span is straightforward, because it's contained in the row, so support for this was available from rowmantic's first version.

However, there is also support for rowspan since version 0.2.0:

Document Result



```
#show table.cell: it => if it.colspan + it.rowspan > 2 { strong(it) } else { it }
#let cell = table.cell
#rowtable(
    separator: ",",
    column-width: 3em, rows: 3em,
    inset: 0.25em,
    [#cell(rowspan: 2, colspan: 2)[C: 2, R: 2], 1, 2, #cell(rowspan: 3)[R: 3], 4],
    [#expandcell[Expandcell], #cell(rowspan: 3)[R: 3]],
    [e, f, g, h],
    [#expandcell[Expandcell], ijk],
)
```