`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[1] on a delimiter which is `&` 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*
goa^{1}	iau^{1}-boe^{3}	koat^{2}-teng^{3}	tang^{7}-si^{5}	boeh^{2}	tng^{1}-khi^{3}
goa^{2}	iau^{2}-boe^{7}	koat^{4}-teng^{7}	tang^{1}-si^{5}	boeh^{4}	tng^{2}-khi^{3}
I	not-yet	decide	when	want	return.

“I have not yet decided when I shall return.”

Input

```
#{
  set table.hline(stroke: 0.08em)
  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: 0pt,      // 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
    [goá   , iáu-boē    , koat-tēng   , tang-sî   , boeh  , tńg-khì    ],
    [goa1  , iau1-boe3  , koat2-teng3 , tang7-si5 , boeh2 , tng1-khi3  ],
    [goa2  , iau2-boe7  , koat4-teng7 , tang1-si5 , boeh4 , tng2-khi3  ],
    [I     , 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 Wikipedia[2]

[1] But shallowly - not looking into styled or nested content

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

Document Result

Term	Explanation	Assumptions
$\mathbf{X}$	Explanatory variables	Non-random
$\mathbf{Y}$	$Y_1, ..., Y_n$ observations	**Pairwise independent**
β	Model parameters	

Input

```
#{
  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(),
    [Term     & Explanation                   & Assumptions ],
    table.hline(),
    [$X$      & Explanatory variables         & Non-random ],
    [$Y$      & $Y_1, ..., Y_n$ observations  & *Pairwise independent*],
    [$beta$   & Model parameters              ],
    table.hline(),
  )
}
```

Trying some more difficult examples

Document Result

Literal &	**Strong**	**X**–*Y*
Equation $\pi = 3.1415...$	$\int_\Omega d\omega$	$X \& Y$
• A • B	1. A 2. B	**A** a **B** b
Figure 1: Top A	See Figure 1 & Figure 2	B Figure 2: Bot
Nested rowtable A B	Nested table A B	`table.cell`
Cell with colspan=2		–
Expandcell		the rest
N/A	N/A	N/A

Input

```
#rowtable(
  align: horizon,
  stroke: 0.1pt,
  row-filler: [N/A],
  [Literal \& & *Strong* & *X*--_Y_ ],
  [Equation \ $pi = 3.1415...$ & $ integral_Omega d omega $  & $X \& Y$],
  [
    - A
    - B
    &
    + 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])
    [&]
    [Nested table \ ]
    table(columns: 2, [A], [B])
    [&]
    table.cell(stroke: 1pt + red)[`table.cell`]
  },
  [#table.cell(fill: yellow.lighten(90%), colspan: 2)[Cell with colspan=2] &
   #table.cell[--]],
  [#expandcell(fill: yellow.lighten(90%))[Expandcell] & #expandcell[the rest]],
  [&&],
  [],
)
```

Double semicolon separator

Document Result

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

Input

```
#rowtable(
  separator: ";;",
  align: horizon,
  stroke: (x, y) => (y: int(y <= 0) * 0.9pt + 0.1pt, x: 0.1pt),
  [First           ;; This is a literal \;\; and ; and , and & ],
  [Second; Third   ;; Equation $pi = 3.1415...$],
  table.hline(stroke: 1pt),
)
```