Term List as a Table

The *term list*[1] is the following syntax in typst:

Input

```
/ term: description
```

Document Result

term description

This package allows laying out the term list as a regular table. By default, the terms form the first column and the descriptions are in the second column, but it can also be transposed. As an extension, additional columns can be added.

Table layout of term description lists is suitable for description lists with few columns, it is not intended as a general replacement for the table function or for all tables.

Examples

Term list as table with no style

Input

```
#terms-table[
  / Package: tabbyterms
  / Technology: Typst
  / Subject: General, Mathematics,
Linguistics
  / Category: Layout, Components
]
```

Document Result

Package	tabbyterms
Technology	Typst
Subject	General, Mathematics, Linguistics
Category	Layout, Components

Term list in plain/default style

Input

```
#show: tabbyterms.style.default-styles
#terms-table[
  / Package: tabbyterms
  / Technology: Typst
  / Subject: General, Mathematics,
Linguistics
  / Category: Layout, Components
]
```

Document Result

Package	tabbyterms
Technology	Typst
Subject	General, Mathematics, Linguistics
Category	Layout, Components

---

[1] https://typst.app/docs/reference/model/terms/

Term list in book style

This assumes the table will have a something suitable as header.

Input

```
#show: tabbyterms.style.default-styles
#terms-table(label: tabbyterms.style.book)[
  / Term: Description
  / Package: tabbyterms
  / Technology: Typst
  / Subject: General, Mathematics,
Linguistics
  / Category: Layout, Components
]
```

Document Result

Term	Description
Package	tabbyterms
Technology	Typst
Subject	General, Mathematics, Linguistics
Category	Layout, Components

Term list transposed[2]

Input

```
#terms-table(transpose: true)[
  / Package: tabbyterms
  / Technology: Typst
]
```

Document Result

Package	Technology
tabbyterms	Typst

Term list with multiple columns

A regular list in the description is expanded into multiple columns. It is not strictly required that the number of entries in each list matches, which makes it easier to edit incrementally.

Input

```
#terms-table(label: tabbyterms.style.book)[
  / Term: - Explanation
          - Assumptions
  / $X$: - Explanatory variables
         - Non-random
  / $Y$: - $Y_1, ..., Y_n$ observations
         - *Pairwise independent*
  / $beta$: - Model parameters
            - Non-random
  / --:
]
```

Document Result

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

Here using the special case syntax `/ --:` to add a `table.hline()` after the last row.

[2]All examples enable default styles if nothing else specified from now on

Style By Label

Labels can be applied by the `label` argument to `terms-table` or directly, as follows.

Input

```
#show: terms-table.with(align: bottom)

/ Cat: #emoji.cat
/ Dog: #emoji.dog
<terms-table-plain>
\

/ Friend: Photo
/ Cat: #emoji.cat
/ Dog: #emoji.dog
/ --:
<terms-table-book>
\

#show <custom-style>: set table(fill:
eastern.lighten(75%), stroke: 0.2pt)
/ Cat: #emoji.cat
/ Dog: #emoji.dog
<custom-style>
\

/ Cat: #emoji.cat
/ Dog: #emoji.dog
<terms-table-revoke>
// revoke inhibits conversion to table
```

Document Result

Cat	🐈
Dog	🐕

Friend	**Photo**
Cat	🐈
Dog	🐕

Cat	🐈
Dog	🐕

Cat 🐈
Dog 🐕

Note that the label attaches naturally to a `terms.item` like this but `terms-table` lifts up the label and attaches it to the whole resulting table.

The *term* and *description* items themselves are tagged with the following labels, allowing them to be styled (and this how the terms are styled with bold in the plain style):

term label: `tabbyterms.style.term`
description label: `tabbyterms.style.description`

Input

```
#show tabbyterms.style.term: emph
#show tabbyterms.style.description:
underline
#show terms: terms-table
/ A: one
/ B: - two
     - three
```

Document Result

A	one	
B	two	three

Glossing Examples

Examples from[3] and[4]. Using a horizontal table version of the term list with no table lines.

```
let gloss-table(body) = {
  set table(stroke: none)
  set table(inset: (left: 0em, right: 1.0em, top: 0.2em, bottom: 0.5em))
  set block(above: 0.65em, below: auto)
  show table.cell.where(y: 0, colspan: 1): emph
  show table.cell.where(colspan: 1): it => {
    if it.y == 0 { return it }
    show regex("[A-Z]{2,}"): smallcaps.with(all: true)
    it
  }
  terms-table(transpose: true, label: <terms-table-gloss>, body)
}
```

Input

```
#show terms: gloss-table

Hittite (Lehmann 1982:211)
/ n=an: CONN=him
/ apedani: that.DAT.SG
/ mehuni: time.DAT.SG
/ essandu: eat.they.shall
/ -table-footer:
  'They shall celebrate him on
that date.' (#smallcaps[conn] =
connective)

Belhare
/ ne-e:
  - DEM-LOC
  - DEM-LOC
/ a-khim-chi:
  - 1SG.POSS-house-PL
  - 1SPOSS-house-PL
/ n-yuNNa:
  - 3NSG-be.NPST
  - 3ns-be.NPST
/ -table-footer:
  'Here are my houses'

Morpheme correspondance example
/ Gila:       now
/ abur-u-n:   they-OBL-GEN
/ ferma:      farm
/ hamišaluğ:  forever
/ güğüna:     behind
/ amuq’-da-č: stay-FUT-NEG
/ -table-footer:
  'Now their farm will not stay
behind forever.'
```

Document Result

Hittite (Lehmann 1982:211)

n=an	*apedani*	*mehuni*	*essandu*
CONN=him	that.DAT.SG	time.DAT.SG	eat.they.shall

‘They shall celebrate him on that date.’ (CONN = connective)

Belhare

ne-e	*a-khim-chi*	*n-yuNNa*
DEM-LOC	1SG.POSS-house-PL	3NSG-be.NPST
DEM-LOC	1SPOSS-house-PL	3ns-be.NPST

‘Here are my houses’

Morpheme correspondance example

Gila	*abur-u-n*	*ferma*	*hamišaluğ*	*güğüna*	*amuq’-da-č*
now	they-OBL-GEN	farm	forever	behind	stay-FUT-NEG

‘Now their farm will not stay behind forever.’

[3] Leipzig Glossing Rules, https://www.eva.mpg.de/lingua/pdf/Glossing-Rules.pdf

[4] Interlinear gloss, https://en.wikipedia.org/wiki/Interlinear_gloss

Function Reference

terms-table

Convert a term list element (`terms`) to a table

By default, the terms are the first column and the descriptions are the second column. Multiple columns can be added by using lists in the descriptions. Table headers and footers are supported. The headers and footers expand to span the width of the whole table, if they consist of just one item.

Additional arguments are forwarded to the `table` function.

Parameters

```
terms-table(
  body: terms content,
  column-width: length array auto,
  header-mark: str,
  footer-mark: str,
  lists-to-columns: bool,
  transpose: bool,
  label: label,
  table: function,
  ..args: arguments
)
```

body `terms` or `content`

Should be a `terms` element or any content where terms should be converted

column-width `length` or `array` or `auto`

Default column width (can also specify regular `columns` argument); if it is an array, extend the array by repeating the last element to cover all columns.

Default: `auto`

header-mark `str`

Name of table header marker row, `none` to disable.

Default: `"-table-header"`

footer-mark `str`

Name of table footer marker row, `none` to disable.

Default: `"-table-footer"`

lists-to-columns `bool`

Whether to expand lists into columns (rows if transposed).

Default: `true`

transpose `bool`

If false, terms and descriptions form separate columns, if true, they form rows.

Default: `false`

label `label`

Which label to apply to the resulting table element

Default: `style.plain`

table `function`

Table function to use to create the table

Default: `std.table`

..args `arguments`

Additional arguments for the table function.

`tabbyterms.style`

default-styles

Template/show rule applying style rules for plain and book styles.

```
#show: tabbyterms.style.default-styles
// the rest of the document
```

Parameters

`default-styles(body: any )`

body `any`

The document

plain

Plain or default style

book

“Booktabs” like style

term

Label for each term in term list as table

description

Label for each description in term list as table

revoke

This label revokes the effects of `terms-table` on the terms.

API Documentation generated using `tidy`.