VarTable

VarTable is a package to make variation table, in a simple way
this package is build on top of fletcher
(version : 0.1.0)

Contents

1 Introduction

this package is designed to simplify the creation of variation tables for functions. To do this, it gives you a typst function, whose parameters are described in detail in this documentation.

a word of warning: it's quite normal that during the array creation process, the elements, such as the lines between the various elements, aren't created as they should be. for example, the line between the labels and the rest, which doesn't go all the way to the end.

if you encounter any bugs, please report them on my GitHub.

2 tabvar function

2.1 general description

tabvar

render a variation table and sign table of your functions

Parameters

```
tabvar(
  init: dictionary,
  domain: array,
  arrow: string,
  content: array,
  stroke: lenght color,
  stroke-arrow: lenght color,
  lign-0: bool
)
```

init `dictionary`

initialitation of the table

- in “variable”, is an content wich contain the table’s variable (like x or t)
- in “label”, you have to put array of 2 arguments that contain in first position the lign’s label and in second position, if the lign is a variation table or a sign table with this following keys : “Variation” and “Sign”

Example : for a variation table of a function f, you should write :

```
init(
  variable: $x$,
  label: (
    ([sign of $f$], "Sign"), //<- the first lign is a sign table
    ([variation of $f$], "Variation") //<- the second lign is a variation table
  )
)
```

Default: (

```
    "variable": [],
    "label": [],
  )
```

domain `array`

values taken by the variable
for example if your funtion changes sign or reaches a max/min for $x \in \{0, 1, 2, 3\}$
you should write this :

```
domain: ($0$, $1$, $2$, $3$)
```

Default: `()`

arrow `string`

the style of the arrow
you can use all diffrents kind of “string” arrow of the package fletcher, so I invite you to read the fletcher documentation

Default: `"->"`

content `array`

the content of the table
see bellow for more details

Default: `((),)`

stroke `lenght` or `color`

the table’s color and thickness
Caution : this stroke can take only lenght or color types but none of the others

Default: `1pt + black`

stroke-arrow `lenght` or `color`

the arrow’s color and thickness
Caution : this stroke can take only lenght or color types but none of the others

Default: `0.6pt + black`

lign-0 `bool`

if you want 0 on lign betwen the sign

Default: `false`

2.2 The content parameter

the content parameter must be an array which must itself contain arrays, as many as there are different labels.

So each of these sub-arrays is equivalent to a line, so there are two cases to distinguish, whether the line corresponds to a sign table or a variation table.

2.2.1 Sign table :

now we call this kind arrays : a sign array
our sign array must be contain as many as there are elements in your domain parameter minus one.

2.2.1.1 A cassical sign array

a sign array must be just contain content like `$+$` or `$-$`, but if you want put anything else like content, you can.

Example :
a normal sign table :

```
#tabvar(
  init: (
    variable: $t$,
    label: (([sign], "Sign"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  content: (($+$, $-$, $+$),),
)
```

t	2	4	6	8
sign	$+$	$-$	$+$	

but if want you can do that :

```
#tabvar(
  init: (
    variable: $t$,
    label: (([sign], "Sign"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  content: (
    (
      "hello world",
      $-$,
      $3/2$
    ),
  ),
)
```

t	2	4	6	8
sign	hello world	$-$	$\frac{3}{2}$	

but I not realy sur about the utility of that
(note : on the second example the table is squeezed with the scale function)

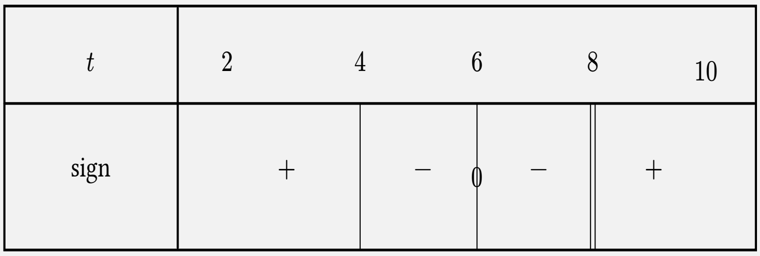
2.2.1.2 Custom separation bar

for all signs except the first, instead of putting the sign directly, you can put a couple, whose first component defines the type of bar just before it.
And there are 3 different types of bar :

- with the `"|"` key, you make a simple bar
- with the `"0"` key, you make a bar with a 0 on the center
- with the `"||"` key, you make a double bar, like for the undefines values

Example :

```
#tabvar(
  init: (
    variable: $t$,
    label: (([sign], "Sign"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  content: (
    (
      $+$,
      ("|", $-$),
      ("0", $-$),
      ("||", $+$)
    ),
  ),
)
```



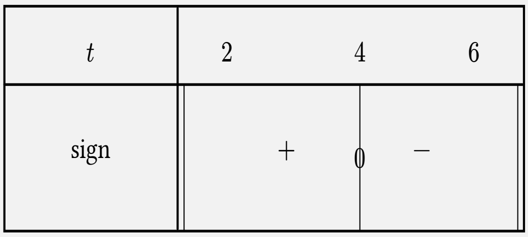
t	2		4		6		8		10
sign		+	|	−	0	−	‖	+	

Note : the `lign-0` parameter is to default lines to `"0"` type or `"|"` type

If you want a double lign at the start, you could, as we have just seen, with the `"||"` type on the very first sign
and at the end, you could add this element `||` at the end of sign array

Example :

```
#tabvar(
  lign-0: true,
  init: (
    variable: $t$,
    label: (([sign], "Sign"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  content: (
    (
      ("||", $+$),
      $-$,
      "||"
    ),
  ),
)
```



t	2		4		6
sign	‖	+	0	−	‖

2.2.1.3 Same sign for more than one value of the variable

for this, is pretty easy, instead of putting the sign directly, you can put a empty couple

Example :

```
#tabvar(
  lign-0: true,
  init: (
    variable: $t$,
    label: (([sign], "Sign"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  content: (
    (
      $+$,
      (),
      $-$
    ),
  ),
)
```

t	2	4	6	8
sign		$+$	0	$-$

2.2.2 Variation table

as for sign array, we'll call them variation array
our sign array must be contain as many as there are elements in your domain parameter.

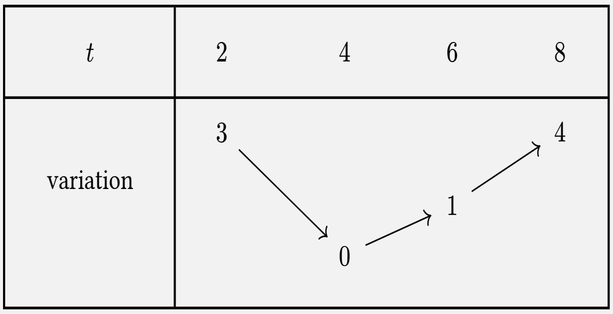
2.2.2.1 A classical variation array

an variation array must be contain couple with in first position, the position; and in second position, whatever you want as long as it's of the content type.

the position can be : `top, center` or `bottom`, but no other type of alignment

Example :

```
#tabvar(
  lign-0: true,
  init: (
    variable: $t$,
    label: (([variation], "Variation"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  content: (
    (
      (top, $3$),
      (bottom, $0$),
      (center, $1$),
      (top, $4$),
    ),
  ),
)
```



2.2.2.2 Undefines values

if your function have certain values undefines like $f(x) = \frac{1}{x}$ for $x = 0$, you certainly want to put a double lign to mean it undefine, and you can!

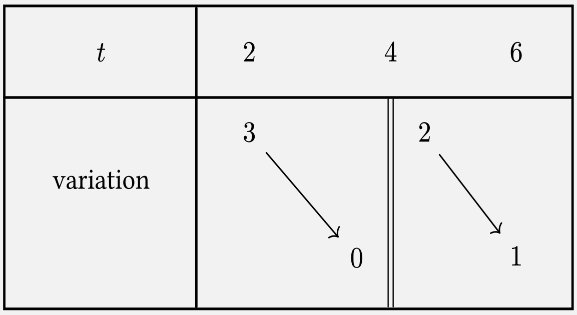
⋆ for each values of domain except the start and the end

the array of one value should look like at `(pos1, pos2, "||", content1, content2)`
where :

- pos1 and 2 is `top, center, bottom` and pos1 is for the placement of content1 similary for pos2
- `"||"` is to precise the value is undefine
- content1 and 2 is type of content and content1 one is for before the double bar and content2 for after

Example :

```
#tabvar(
  lign-0: true,
  init: (
    variable: $t$,
    label: (([variation], "Variation"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  content: (
    (
      (top, $3$),
      (bottom, top, "||", $0$, $2$),
      (bottom, $1$),
    ),
  ),
)
```

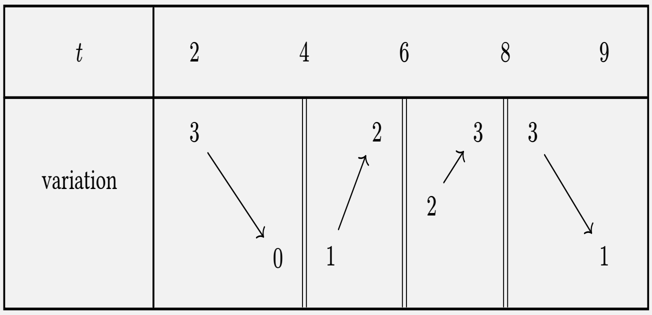


if `pos1` and `pos2` is same, you can just fill in one instead of two,
in the same way if `content1` and `content2` is same, you can also enter just one

Example :

instead of `(top, top, "||" , $0$, $0$)` you can use `(top, "||" ,$0$)`

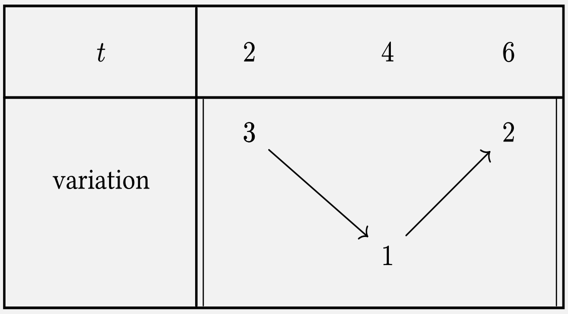
```
#tabvar(
  lign-0: true,
  init: (
    variable: $t$,
    label: (([variation], "Variation"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  content: (
      (
        (top, $3$),
        (bottom, "||", $0$, $1$),
        (top, center, "||", $2$),
        (top, "||", $3$),
        (bottom, $1$),
      ),
  ),
)
```



⋆ for the first and the end values

it a basic array but with `"||"` this parameter at the array's center
for example `(top, "||", $3$)`

```
#tabvar(
  lign-0: true,
  init: (
    variable: $t$,
    label: (([variation], "Variation"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  content: (
      (
        (top, "||", $3$),
        (bottom, $1$),
        (top, "||", $2$),
      ),
  ),
)
```



2.2.2.3 to skip a value