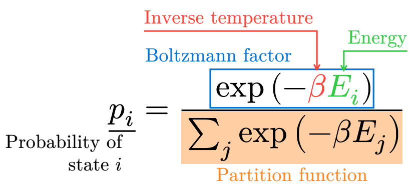
mannot v0.2.0

A package for marking and annotating in math blocks in Typst.

Contents

Example



```
#set text(12pt)
#v(2em)
$
  markul(p_i, tag: #<p>)
  = markrect(
    exp(- marktc(beta, tag: #<beta>) marktc(E_i, tag: #<E>, color: #green)),
    tag: #<Boltzmann>, color: #blue,
  ) / mark(sum_j exp(- beta E_j), tag: #<Z>)

  #annot(<p>, pos: left)[Probability of \ state $i$]
  #annot(<beta>, pos: top + left, yshift: 2em)[Inverse temperature]
  #annot(<E>, pos: top + right, yshift: 1em)[Energy]
  #annot(<Boltzmann>, pos: top + left)[Boltzmann factor]
  #annot(<Z>)[Partition function]
$
```

Usage

Import the package `mannot` on the top of your document:

```
#import "@preview/mannot:0.2.0": *
```

To define the target of an annotation within a math block, use the following marking functions:

- `mark`: marks the content with highlighting;
- `markrect`: marks the content with a rectangular box;
- `markul`: marks the content with an underline;
- `marktc`: marks the content and changes the text color.

```
$
mark(x, tag: #<t1>) + markrect(2y, tag: #<t2>)
+ markul(z+1, tag: #<t3>) + marktc(C, tag: #<t4>)
$
```

$$x + \boxed{2y} + \underline{z + 1} + C$$

You can customize the marking color and other styles:

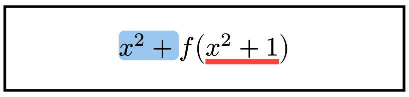
```
$
mark(x, tag: #<t1>, color: #purple)
```

$$x + \boxed{2y} + \underline{z + 1} + C$$

```
+ markrect(2y, tag: #<t2>, color: #red, padding: #2pt)
+ markul(z+1, tag: #<t3>, stroke: #1pt)
+ marktc(C, tag: #<t4>, color: #olive)
$
```

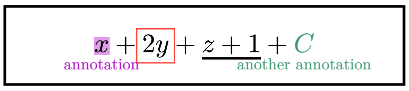
You can also use marking functions solely for styling parts of a math block, without tags:

```
$
mark(x^2 +, color: #blue, radius: #20%)
f(markul(x^2 + 1, color: #red, stroke: #2pt))
$
```



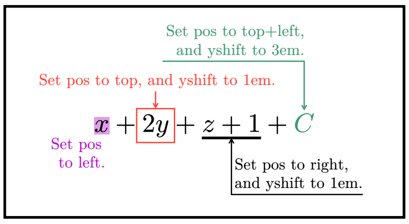
Once you have marked content with a tag, you can annotate it using the `annot` function within the same math block:

```
$
mark(x, tag: #<t1>, color: #purple)
+ markrect(2y, tag: #<t2>, color: #red, padding: #2pt)
+ markul(z+1, tag: #<t3>, stroke: #1pt)
+ marktc(C, tag: #<t4>, color: #olive)

#annot(<t1>)[annotation]
#annot(<t4>)[another annotation]
$
```



You can customize the position of the annotation and its vertical distance from the marked content, using the `pos` and `yshift` parameters of the `annot` function:

```
#v(3em)
$
mark(x, tag: #<t1>, color: #purple)
+ markrect(2y, tag: #<t2>, color: #red, padding: #2pt)
+ markul(z+1, tag: #<t3>, stroke: #1pt)
+ marktc(C, tag: #<t4>, color: #olive)

#annot(<t1>, pos: left)[Set pos \ to left.]
#annot(<t2>, pos: top, yshift: 1em)[
  Set pos to top, and yshift to 1em.
]
#annot(<t3>, pos: right, yshift: 1em)[
  Set pos to right,\ and yshift to 1em.
]
#annot(<t4>, pos: top + left, yshift: 3em)[
  Set pos to top+left,\ and yshift to 3em.
]
$
#v(2em)
```



Limitations

If you mark a inline math element containing linebreaks, its layout will be broken:

```
$mark(x + x + x + x + x + x + x + x)$
```

$$x + x + x + x + x + x + x + x$$

API

- core-mark()
- mark()
- markrect()
- markul()
- marktc()
- core-annot()
- annot()

core-mark

Marks content within a math block with a custom underlay or overlay.

This function measures the position and size of the marked content, applies a custom underlay or overlay, and generates metadata associated with a given tag. The metadata includes the original content, its position (`x`, `y`), dimensions (`width`, `height`), and the color used for the marking. This metadata can be later used for annotations.

Use this function as a foundation for defining custom marking functions.

Example:

```
#let mymark(body, tag: none) = {
  let overlay(width, height, color) = {
    rect(width: width, height: height, stroke: color)
  }
  return core-mark(body, tag: tag, color: red, overlay: overlay, padding: (y: .1em))
}

$ mymark(x, tag: #<e>) $

#context {
  let info = query(<e>).last()
  repr(info.value)
}
```

$$\boxed{x}$$

```
(
  body: [x],
  x: 446.6pt,
  y: 540.36pt,
  width: 6.1pt,
  height: 6.97pt,
  color: rgb("#ff851b"),
)
```

Parameters

```
core-mark(
  body: content,
  tag: none label,
  color: color,
  underlay: none function,
  overlay: none function,
  padding: none length dictionary,
  sizestyle: auto string
) -> content
```

body `content`

The content to be marked within a math block.

tag `none` or `label`

An optional tag to associate with the metadata. This tag can be used to query the marked element.

Default: `none`

color `color`

The color used for marking.

Default: `black`

underlay `none` or `function`

An optional function to create a custom underlay. This function receives the marked content's width and height (including padding) and marking color, and should return content to be placed **under** the marked content. The signature is `underlay(width, height, color)`.

Default: `none`

overlay `none` or `function`

An optional function to create a custom overlay. This function receives the marked content's width and height (including padding) and marking color, and should return content to be placed **over** the marked content. The signature is `overlay(width, height, color)`.

Default: `none`

padding `none` or `length` or `dictionary`

The spacing between the marked content and the edge of the underlay/overlay. This can be specified as a single `length` value, which applies to all sides, or as a `dictionary` of `length` with keys `left`, `right`, `top`, `bottom`, `x`, `y`, or `rest`.

Default: `(:)`

sizestyle `auto` or `string`

The math size style of the marked content. Possible values are `"display"`, `"inline"`, `"script"`, and `"sscript"`. This is used for measuring the content's height. If set to `auto`, the style is determined automatically based on its width.

Default: `auto`

mark

Marks content within a math block with highlighting.

If you mark content with a tag, you can annotate it using the `annot` function.

Example

```
$ mark(x) $
```

x

Parameters

```
mark(
  body: content,
  tag: none label,
  color: auto color,
  fill: auto none color gradient pattern,
  stroke: none auto length color gradient stroke pattern dictionary,
  radius: relative dictionary,
  padding: none length dictionary,
  sizestyle: auto string
) -> content
```

body `content`

The content to be highlighted within a math block.

tag `none` or `label`

An optional tag used to identify the marked content for later annotations.

Default: `none`

color `auto` or `color`

The color used for the highlight and later annotations. If both `color` and `fill` are set to `auto`, `color` defaults to `orange`. Otherwise, if only `color` is `auto`, it defaults to `black`.

Default: `auto`

fill `auto` or `none` or `color` or `gradient` or `pattern`

The fill style for the highlight rectangle. If set to `auto`, `fill` will be set to `color.transparentize(60%)`.

Default: `auto`

stroke `none` or `auto` or `length` or `color` or `gradient` or `stroke` or `pattern` or `dictionary`

The stroke style for the highlight rectangle.

Default: `none`

radius `relative` or `dictionary`

The corner radius of the highlight rectangle.

Default: `(:)`

padding `none` or `length` or `dictionary`

The spacing between the marked content and the edge of the highlight rectangle. This can be specified as a single `length` value which applies to all sides, or as a `dictionary` of `length` with keys `left`, `right`, `top`, `bottom`, `x`, `y`, or `rest`.

Default: `(y: .1em)`

sizestyle `auto` or `string`

The math size style of the marked content. Possible values are `"display"`, `"inline"`, `"script"` and `"sscript"`. If set to `auto`, the style is determined automatically.

Default: `auto`

markrect

Marks content within a math block with a rectangle.

If you mark content with a tag, you can annotate it using the `annot` function.

Example

```
$ markrect(x + y) $
```

$$\boxed{x + y}$$

Parameters

```
markrect(
  body: content,
  tag: none label,
  color: color,
  fill: none color gradient pattern,
  stroke: none length color gradient stroke pattern dictionary,
  radius: relative dictionary,
  padding: none length dictionary,
  sizestyle: auto string
) -> content
```

body `content`

The content to be marked within a math block.

tag `none` or `label`

An optional tag used to identify the content for later annotations.

Default: `none`

color `color`

The color used for the rectangle stroke and later annotations.

Default: `black`

fill `none` or `color` or `gradient` or `pattern`

The fill style for the rectangle.

Default: `none`

stroke `none` or `length` or `color` or `gradient` or `stroke` or `pattern` or `dictionary`

The stroke style for the rectangle. If its `paint` is set to `auto`, it will be set to the `color`.

Default: `.05em`

radius `relative` or `dictionary`

The corner radius of the rectangle.

Default: `(:)`

padding `none` or `length` or `dictionary`

The spacing between the marked content and the edge of the rectangle. This can be specified as a single `length` value which applies to all sides, or as a `dictionary` of `length` with keys `left`, `right`, `top`, `bottom`, `x`, `y`, or `rest`.

Default: `(y: .1em)`

sizestyle `auto` or `string`

The math size style of the marked content. Possible values are `"display"`, `"inline"`, `"script"` and `"sscript"`. If set to `auto`, the style is determined automatically.

Default: `auto`

markul

Marks content within a math block with an underline.

If you mark content with a tag, you can annotate it using the `annot` function.

Example

```
$ markul(x + y) $
```

$$\underline{x + y}$$

Parameters

```
markul(
  body: content,
  tag: none label,
  color: color,
  stroke: none length color gradient stroke pattern dictionary,
  padding: none length,
  sizestyle
) -> content
```

body `content`

The content to be underlined within a math block.

tag `none` or `label`

An optional tag used to identify the underlined content for later annotations.

Default: `none`

color `color`

The color used for the underline.

Default: `black`

stroke `none` or `length` or `color` or `gradient` or `stroke` or `pattern` or `dictionary`

The stroke style for the underline.

Default: `.05em`

padding `none` or `length`

The spacing between the marked content and the underline.

Default: `.15em`

sizestyle

The math size style of the marked content. Possible values are `"display"`, `"inline"`, `"script"` or `"sscript"`. If set to `auto`, the style is determined automatically.

Default: `auto`

marktc

Marks content within a math block and changes its text color.

If you mark content with a tag, you can annotate it using the `annot` function.

Example

```
$ marktc(x + y) $
```

$$x + y$$

Parameters

```
marktc(
  body: content,
  tag: none label,
  color: color,
  sizestyle
) -> content
```

body `content`

The content to be underlined within a math block.

tag `none` or `label`

An optional tag used to identify the underlined content for later annotations.

Default: `none`

color `color`

The color used for the underline.

Default: `red`

sizestyle

The math size style of the marked content. Possible values are `"display"`, `"inline"`, `"script"` or `"sscript"`. If set to `auto`, the style is determined automatically.

Default: `auto`

core-annot

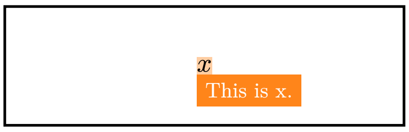
Places a custom annotation on previously marked content within a math block.

This function creates a custom annotation by applying an overlay to content that was previously marked with a specific tag using `core-mark`. It must be used within the same math block as the marked content.

Use this function as a foundation for defining custom annotation functions.

Example

```
#let myannot(tag, annotation) = {
  let overlay(width, height, color) = {
    set text(white, .8em)
    let annot-block = box(fill: color, inset: .4em,
annotation)
    place(annot-block, dy: height)
  }
```



```
    return core-annot(tag, overlay)
}

$
mark(x, tag: #<e>)
#myannot(<e>)[This is x.]
$
```

Parameters

```
core-annot(
  tag: label,
  overlay: function
) -> content
```

tag `label`

The tag associated with the content to annotate. This tag must match a tag previously used in a `core-mark` call.

overlay `function`

The function to create the custom annotation overlay. This function receives the width, height, and color of the marked content (including padding) and should return content to be placed **over** the marked content. The signature is `overlay(width, height, color)`.

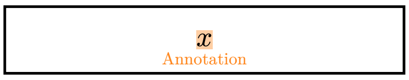
annot

Places an annotation on previously marked content within a math block.

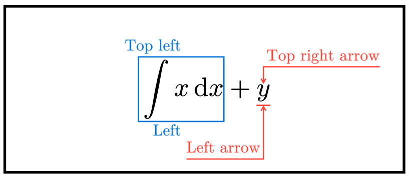
This function must be used within the same math block as the marked content.

Example

```
$
mark(x, tag: #<e>)
#annot(<e>)[Annotation]
$
```



```
$
markrect(integral x dif x, tag: #<x>, color: #blue)
+ markul(y, tag: #<y>, color: #red)

#annot(<x>, pos: left)[Left]
#annot(<x>, pos: top + left)[Top left]
#annot(<y>, pos: left, yshift: 2em)[Left arrow]
#annot(<y>, pos: top + right, yshift: 1em)[Top right
arrow]
$
```



Parameters

```
annot(
  tag: label ,
  annotation: content ,
  pos: alignment ,
  yshift: length ,
  text-props: dictionary ,
  par-props: dictionary ,
  alignment: auto  alignment ,
  show-arrow: auto  bool ,
  arrow-stroke: auto  none  color  length  dictionary  stroke ,
  arrow-padding: length
) -> content
```

tag `label`

The tag associated with the content to annotate. This tag must match a tag previously used in marking.

annotation `content`

The content of the annotation.

pos `alignment`

The position of the annotation relative to the marked content. Possible values are (`top` or `bottom`) + (`left`, `center` or `right`).

Default: `center + bottom`

yshift `length`

The vertical offset between the annotation and the marked content.

Default: `.2em`

text-props `dictionary`

Properties for the annotation text. If the `fill` property is not specified, it defaults to the marking's color.

Default: `(size: .6em, bottom-edge: "descender")`

par-props `dictionary`

Properties for the annotation paragraph.

Default: `(leading: .3em)`

alignment `auto` or `alignment`

The alignment of the annotation text within its box.

Default: `auto`

show-arrow `auto` or `bool`

Whether to display an arrow connecting the annotation to the marked content. If set to `auto`, an arrow is shown when `yshift` is greater than `0.4em`.

Default: `auto`

arrow-stroke `auto` or `none` or `color` or `length` or `dictionary` or `stroke`

The stroke style for the arrow. If the `paint` property is not specified, it defaults to the marking's color.

Default: `.08em`

arrow-padding `length`

The spacing between the arrow and the annotation box.

Default: `.2em`