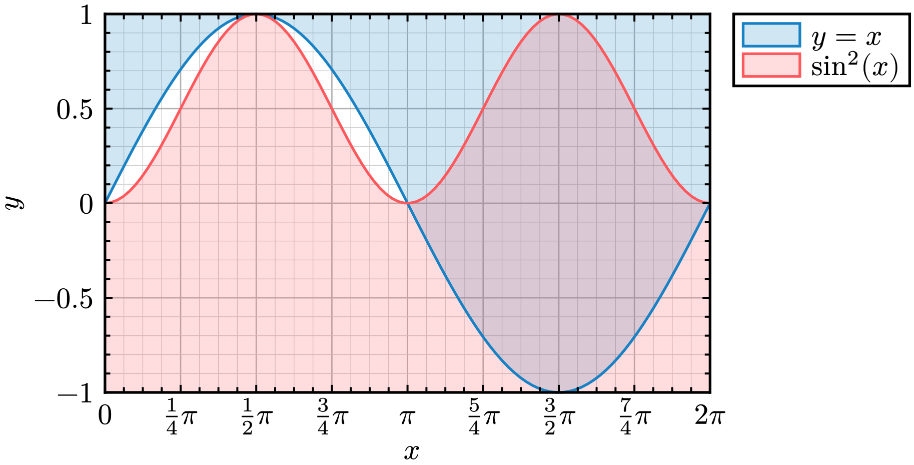
CeTZ Plot

Johannes Wolf
fenjalien

Version 0.1.0



1 Introduction

CeTZ-Plot is a package for making plots in Typst using CeTZ.

2 Usage

This is the minimal starting point:

```
#import "@preview/cetz:0.2.2"
#import "@preview/cetz-plot:0.1.0"
#cetz.canvas({
  cetz-plot.plot(...,{

  })
})
```

Note that plot functions are imported inside the scope of the `canvas` block. All following example code is expected to be inside a `canvas` block, with the `cetz-plot` module imported into the namespace.

3 Plot

3.1 plot

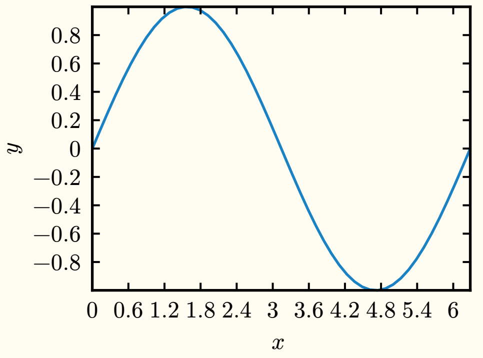
Create a plot environment. Data to be plotted is given by passing it to the `plot.add` or other plotting functions. The plot environment supports different axis styles to draw, see its parameter `axis-style:`.

3.1.1 Parameters

```
plot(
  body: body ,
  size: array ,
  axis-style: axis-style-module ,
  name: string  none ,
  plot-style: style  function ,
  mark-style: style  function ,
  legend: none  auto  coordinate ,
  legend-anchor: auto  string ,
  legend-style: style ,
  ..options: any
)
```

body: `body`

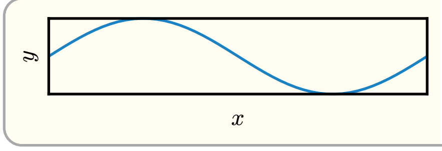
Calls of `plot.add` or `plot.add-*` commands. Note that normal drawing commands like `line` or `rect` are not allowed inside the plots body, instead wrap them in `plot.annotate`, which lets you select the axes used for drawing.



```
cetz-plot.plot({
  cetz-plot.add.xy(calc.sin, domain: (0,2*calc.pi))
})
```

size: `array` Default: `(5,5 * 3/4)`

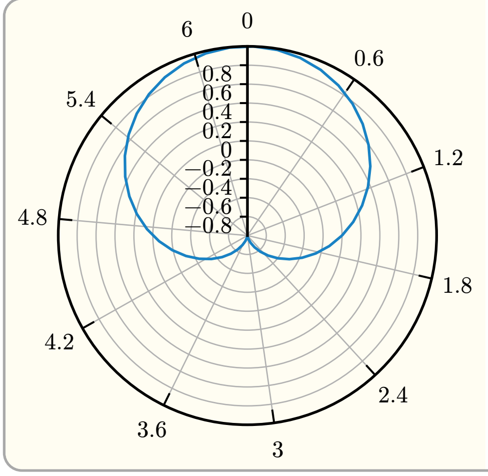
Plot size tuple of `(<width>, <height>)` in canvas units. This is the plots inner plotting size without axes and labels. this value, as it doesn't include axis labels, ticks, or the legend.



```
cetz-plot.plot(
  size: (5,1),
  x-tick-step: none, y-tick-step: none,
  {cetz-plot.add.xy(calc.sin, domain: (0,2*calc.pi))}
)
```

axis-style: `axis-style-module` Default: `axis-style.orthorect-2d`

TODO: Make this link to the axis-style section



```
cetz-plot.plot(
  size: (5,5),
  axis-style: cetz-plot.axis-style.polar-2d,
  x-grid: "both", y-grid: "both",
  {cetz-plot.add.xy(calc.sin, domain: (0,2*calc.pi))}
)
```

name: `string` or `none` Default: `none`

The plots element name to be used when referring to anchors

plot-style: `style` or `function` Default: `default-plot-style`

Styling to use for drawing plot graphs. This style gets inherited by all plots and supports `palette` functions. The following style keys are supported:

stroke: `none` or `stroke` Default: `1pt`

Stroke style to use for stroking the graph.

fill: `none` or `paint` Default: `none`

Paint to use for filled graphs. Note that not all graphs may support filling and that you may have to enable filling per graph, see `plot.add(fill: ..)`.

mark-style: `style` or `function` Default: `default-mark-style`

Styling to use for drawing plot marks. This style gets inherited by all plots and supports `palette` functions. The following style keys are supported:

stroke: `none` or `stroke` Default: `1pt`

Stroke style to use for stroking the mark.

fill: `none` or `paint` Default: `none`

Paint to use for filling marks.

legend: `none` or `auto` or `coordinate` Default: `auto`

The position the legend will be drawn at. See plot-legends for information about legends. If set to `<auto>`, the legend's "default-placement" styling will be used. If set to a `<coordinate>`, it will be taken as relative to the plot's origin.

legend-anchor: `auto` or `string` Default: `auto`

Anchor of the legend group to use as its origin. If set to `auto` and `lengend` is one of the predefined legend anchors, the opposite anchor to `legend` gets used.

legend-style: `style` Default: `(:)`

Style key-value overwrites for the legend style with style root `legend`.

..options: `any`

Axis options, see *options* below.

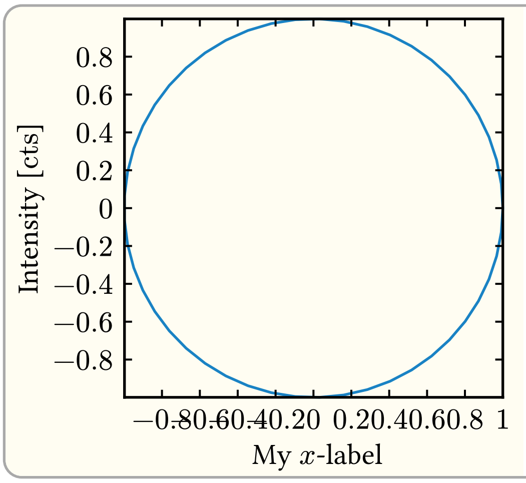
To draw elements insides a plot, using the plots coordinate system, use the `plot.annotate(..)` function.

3.1.2 Options

You can use the following options to customize each axis of the plot. You must pass them as named arguments prefixed by the axis name followed by a dash (`-`) they should target. Example: `x-min: 0`, `y-ticks: (..)` or `x2-label: [..]`.

label: `none` or `content` Default: `none`

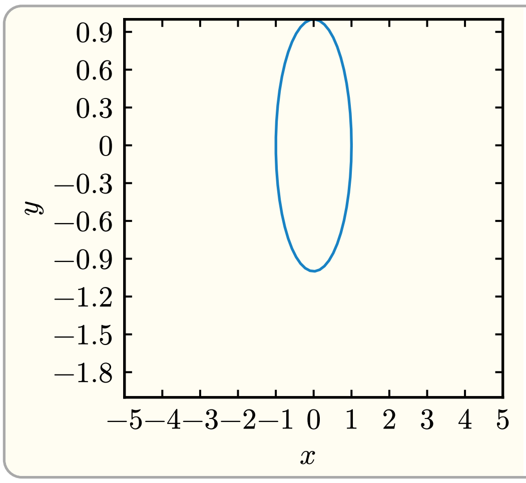
The axis' label. If and where the label is drawn depends on the `axis-style`.



```
cetz-plot.plot(
  size: (5,5),
  x-label: [My $x$-label],
  y-label: [Intensity [$"cts"$]],
  {
    cetz-plot.add.xy(
      domain: (0, 2 * calc.pi),
      t => (calc.cos(t), calc.sin(t))
    )
  }
)
```

min: `auto` or `float` Default: `auto`

Axis lower domain value. If this is set greater than than `max`, the axis' direction is swapped



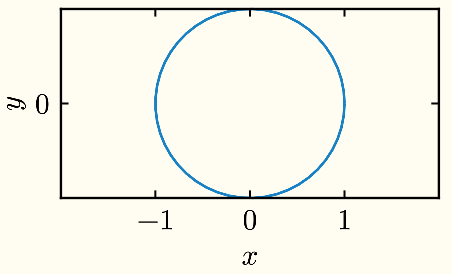
```
cetz-plot.plot(
  size: (5,5),
  x-min: -5, x-max: 5,
  y-min: -2,
  {
    cetz-plot.add.xy(
      domain: (0, 2 * calc.pi),
      t => (calc.cos(t), calc.sin(t))
    )
  }
)
```

max: `auto` or `float` Default: `auto`

Axis upper domain value. If this is set to a lower value than `min`, the axis' direction is swapped

equal: `string` Default: `none`

Set the axis domain to keep a fixed aspect ratio by multiplying the other axis domain by the plots aspect ratio, depending on the other axis orientation (see `horizontal`). This can be useful to force one axis to grow or shrink with another one. You can only "lock" two axes of different orientations.



```
cetz-plot.plot(
  size: (5,2.5),
  x-tick-step: 1, y-tick-step: 1,
  x-equal: "y",
  {
    cetz-plot.add.xy(
      domain: (0, 2 * calc.pi),
      t => (calc.cos(t), calc.sin(t))
    )
  }
)
```

horizontal: `bool` Default: `"axis name dependant"`

If true, the axis is considered an axis that gets drawn horizontally, vertically otherwise. The default value depends on the axis name on axis creation. Axes which name start with `x` have this set to `true`, all others have it set to `false`. Each plot has to use one horizontal and one vertical axis for plotting, a combination of two y-axes will panic: ("y", "y2").

tick-step: `none` or `auto` or `float` Default: `auto`

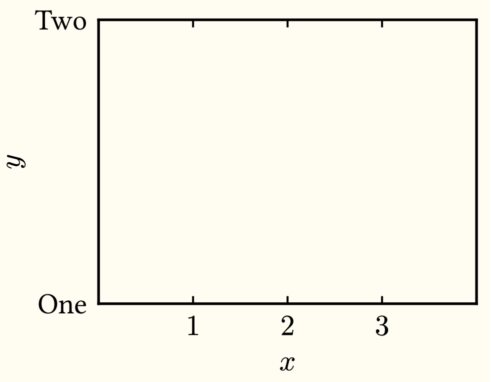
The increment between tick marks on the axis. If set to `auto`, an increment is determined. When set to `none`, incrementing tick marks are disabled.

minor-tick-step: `none` or `float` Default: `none`

Like `tick-step`, but for minor tick marks. In contrast to ticks, minor ticks do not have labels.

ticks: `none` or `array` Default: `none`

A List of custom tick marks to additionally draw along the axis. They can be passed as an array of `<float>` values or an array of `(<float>, <content>)` tuples for setting custom tick mark labels per mark.



```
cetz-plot.plot(
  x-min: 0, x-max: 4,
  x-tick-step: none,
  x-ticks: (1, 2, 3),

  y-min: 1, y-max: 2,
  y-tick-step: none,
  y-ticks: ((1, [One]), (2, [Two])),
  {
    cetz-plot.add.xy(((0,0),))
  }
)
```

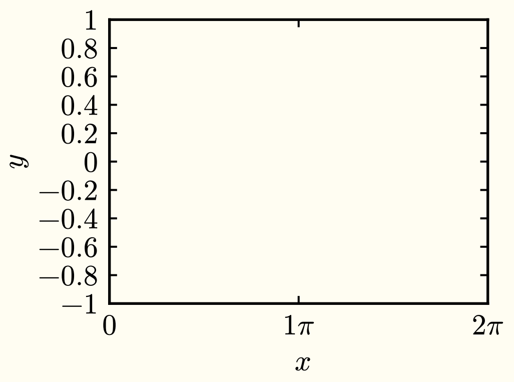
Examples: `(1, 2, 3)` or `((1, [One]), (2, [Two]), (3, [Three]))`

format: `none` or `string` or `function` Default: `"float"`

How to format the tick label: You can give a function that takes a `<float>` and return `<content>` to use as the tick label. You can also give one of the predefined options:

float Floating point formatting rounded to two digits after the point (see `decimals`)

sci Scientific formatting with × 10^n used as exponet syntax



```
let formatter(v) = if v != 0 {
  $ #{v/calc.pi} pi $
} else {
  $ 0 $
}

cetz-plot.plot(
  x-tick-step: calc.pi,
  x-min: 0, x-max: 2 * calc.pi,
  x-format: formatter,
{
  cetz-plot.add.xy(((0,0),))
})
```

decimals: `int` Default: `2`

Number of decimals digits to display for tick labels, if the format is set to `"float"`.

unit: `none` or `content` Default: `none`

Suffix to append to all tick labels.

mode: `none` or `string` Default: `none`

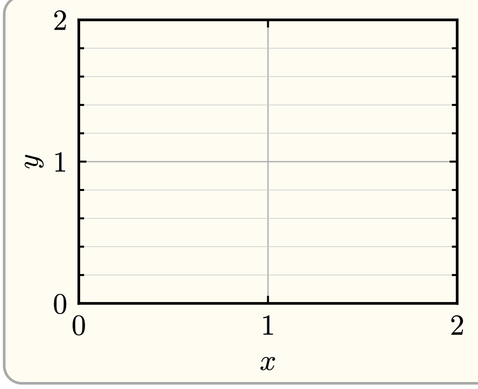
The scaling function of the axis. Takes `lin` (default) for linear scaling, and `log` for logarithmic scaling.

base: `none` or `number` Default: `none`

The base to be used when labeling axis ticks in logarithmic scaling

grid: `bool` or `string` Default: `false`

If `true` or `"major"`, show grid lines for all major ticks. If set to `"minor"`, show grid lines for minor ticks only. The value `"both"` enables grid lines for both, major- and minor ticks.



```
cetz-plot.plot(
  x-min: 0, x-max: 2, x-grid: "both",
  x-tick-step: 1,

  y-min: 0, y-max: 2, y-grid: "both",
  y-tick-step: 1, y-minor-tick-step: .2,
  {
    cetz-plot.add.xy(((0,0),))

  }
)
```

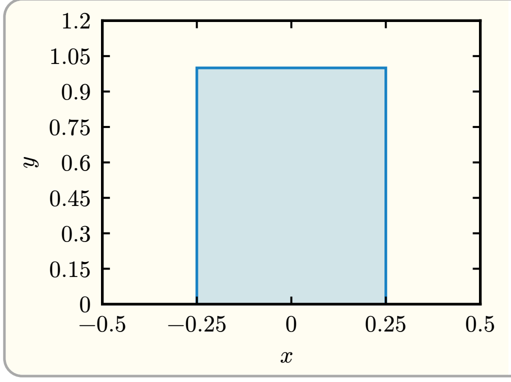
break: `bool` Default: `false`

If true, add a “sawtooth” at the start or end of the axis line, depending on the axis bounds. If the axis min. value is > 0, a sawtooth is added to the start of the axes, if the axis max. value is < 0, a sawtooth is added to its end.

3.2 Add

3.2.1 bar

Adds a series of bars. Bars are of `bar-width` total width, centered at a given `x` coordinate, between heights `y-offset` (default: `0`) and `y-offset` + `y`.



```
cetz-plot.plot(
  x-min: -0.5, x-max: 0.5, x-tick-step: 0.25,
  y-max: 1.2,
 {
  cetz-plot.add.bar(
    ((0,1),),
    bar-width: 0.5,
  )
})
```

Parameters

```
bar(
  data: array,
  x-key: string int,
  y-key: string int,
  y-offset-key: string int,
  bar-width: float,
  label: content,
  style: style,
  axes: axes
)
```

data: `array`

An array representing a single series of bars. Entries can be of type `array` or `dictionary`, and must contain within them an `x` coordinate, and optionally a `y` coordinate expressing the magnitude of the bar to add, and optionally a `y-offset` coordinate (default: `0`) which dictates where the bar's base is draw.

x-key: `string` or `int` Default: `0`

The key at which the `x` coordinate is described in each `data` entry.

y-key: `string` or `int` Default: `1`

The key at which the `y` coordinate is described in each `data` entry.

y-offset-key: `string` or `int` Default: `none`

The key at which the `y-offset` coordinate is described in each `data` entry. If `none`, the `y-offset` is assumed to be `0` for each entry. If `y-offset-key` is not contained within an entry despite being set, the `y-offset` is assumed to be `0`.

bar-width: `float` Default: `0.5`

The width of the bar along the `x` axis, in data-viewport space. The bar is drawn centered about its `x` coordinate, therefore, the bar extends by `bar-width`/2 either side.

label: `content` Default: `none`

The label to be shown in the legend. If `none`, no entry is shown in the legend.

style: `style` Default: `(:)`

Style to use, can be used with a `palette` function

axes: `axes` Default: `("x", "y")`

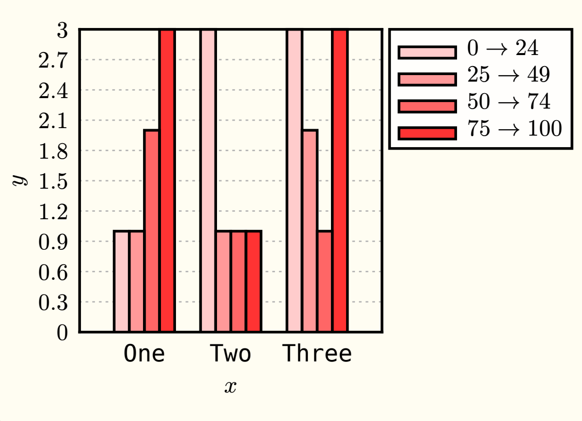
Name of the axes to use for plotting. Reversing the axes means rotating the plot by 90 degrees.

4 Chart

4.1 Bar

4.1.1 clustered

Render a clustered bar chart



```
cetz-plot.chart.bar.clustered(
  size: (4,4),
  (
    ([One],   1, 1, 2, 3),
    ([Two],   3, 1, 1 ,1),
    ([Three], 3, 2, 1, 3),
  ),
  label-key: 0,
  y-keys: (1,2,3,4),
  labels: (
    $0 -> 24$,
    $25 -> 49$,
    $50 -> 74$,
    $75 -> 100$
  ),
)
```

Parameters

```
clustered(
  data,
  labels,
  label-key,
  y-keys,
  y-error-keys,
  bar-width,
  bar-spacing,
  bar-style,
  axes,
  ..plot-args
)
```

data:

labels: Default: `()`

label-key: Default: `0`

y-keys: Default: `(1,)`

y-error-keys: Default: `none`

bar-width: Default: `0.7`

bar-spacing: Default: `0`

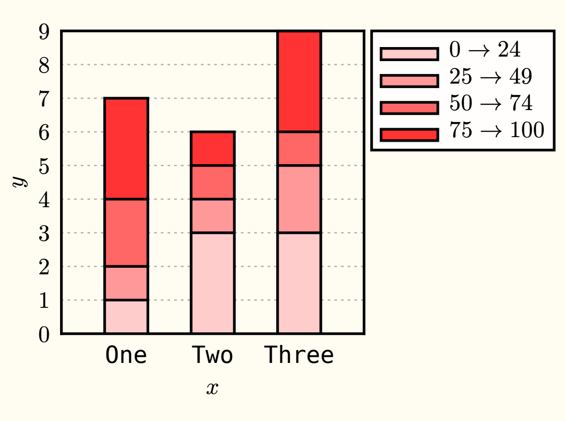
bar-style: Default: `palette.red`

axes: Default: `("x", "y")`

..plot-args:

4.1.2 stacked

Render a stacked bar chart



```
cetz-plot.chart.bar.stacked(
  size: (4,4),
  (
    ([One],   1, 1, 2, 3),
    ([Two],   3, 1, 1 ,1),
    ([Three], 3, 2, 1, 3),
  ),
  label-key: 0,
  y-keys: (1,2,3,4),
  labels: (
    $0 -> 24$,
    $25 -> 49$,
    $50 -> 74$,
    $75 -> 100$
  ),
)
```

Parameters

```
stacked(
  data,
  labels,
  label-key,
  y-keys,
  y-error-keys,
  bar-width,
  bar-style,
  axes,
  ..plot-args: variadic
)
```

data:

labels: Default: `()`

label-key: Default: `0`

y-keys: Default: `(1,)`

y-error-keys: Default: `none`

bar-width: Default: `0.5`

bar-style: Default: `palette.red`

axes: Default: `("x", "y")`

..plot-args: `variadic`

Additional plotting parameters and axis options to be passed to `plot()`