simple-plot

Typst Package

Mathematical Function Plotting

A lightweight library for creating elegant mathematical plots

Version 0.3.0

Nathan Scheinmann

Contents

1. Introduction

`simple-plot` is a Typst package for creating clean, elegant mathematical plots. Built on CeTZ, it provides an intuitive interface for plotting functions, data points, and creating publication-ready graphs.

1.1. Features

- Plot mathematical functions with automatic sampling
- Scatter plots and line plots with customizable markers
- Clean integer-based tick system by default
- Major and minor grid with elegant styling
- Gap-based grid line breaks around tick labels (grid-label-break)
- Automatic axis extension beyond grid
- Flexible axis positioning (origin, bottom/left, custom)
- Multiple label display options (unit-label-only, label-step)
- Function labels with flexible positioning
- Clipping for clean rendering at boundaries

1.2. Installation

Import the package in your Typst document:

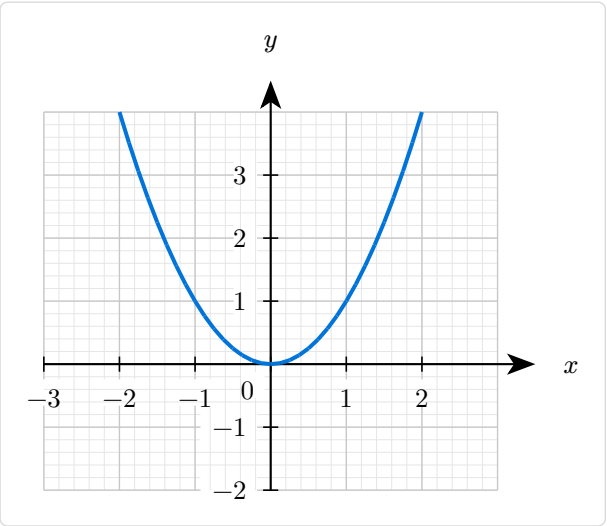
```
#import "@local/simple-plot:0.3.0": plot
```

1.3. Quick Start

Code

```
#plot(
  width: 6, height: 5,
  xmin: -3, xmax: 3, ymin: -2, ymax: 4,
  xlabel: $x$, ylabel: $y$,
  show-grid: true,
  (fn: x => x * x, stroke: blue + 1.5pt),
)
```

Preview



2. Basic Usage

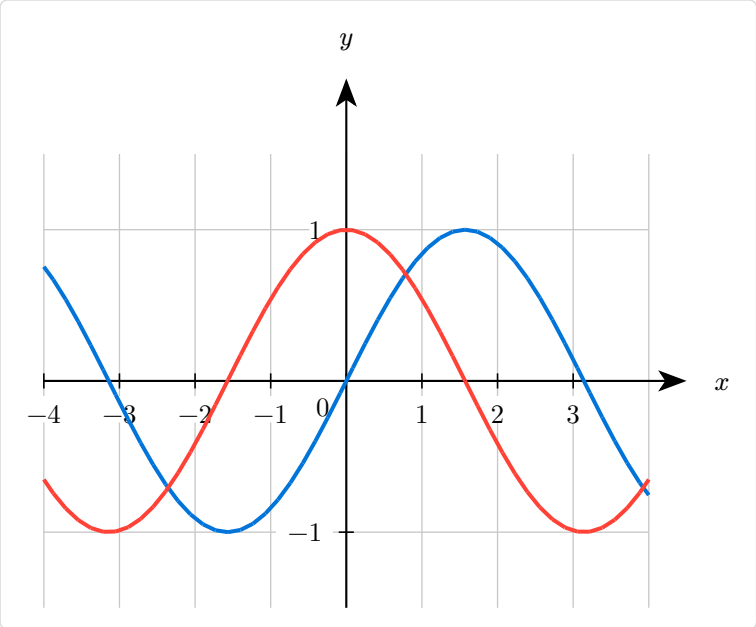
2.1. Plotting Functions

Plot mathematical functions by passing a dictionary with `fn`:

Code

```
#plot(
  width: 8, height: 6,
  xmin: -4, xmax: 4, ymin: -1.5, ymax: 1.5,
  xlabel: $x$, ylabel: $y$,
  show-grid: "major",
  (fn: x => calc.sin(x), stroke: blue + 1.5pt),
  (fn: x => calc.cos(x), stroke: red + 1.5pt),
)
```

Preview



2.2. Mathematical Functions Reference

Functions are defined using Typst's `calc` module. Here are the most common mathematical functions:

Function	Typst syntax
Power x^n	`calc.pow(x, n)`
Square root $\sqrt{x}$	`calc.sqrt(x)`
Absolute value $\|x\|$	`calc.abs(x)`
Sine $\sin(x)$	`calc.sin(x)`
Cosine $\cos(x)$	`calc.cos(x)`
Tangent $\tan(x)$	`calc.tan(x)`
Exponential e^x	`calc.exp(x)`
Natural log $\ln(x)$	`calc.ln(x)`
Log base b	`calc.log(x, base: b)`
Maximum	`calc.max(a, b)`
Minimum	`calc.min(a, b)`

Important: When using constants in calculations, use decimal notation (e.g., `2.0` instead of `2`) to avoid type errors. For example:

- ✓ `x => x * x / 2.0`
- ✗ `x => x * x / 2` *(may cause errors)*

This is because Typst's type system requires consistent float arithmetic.

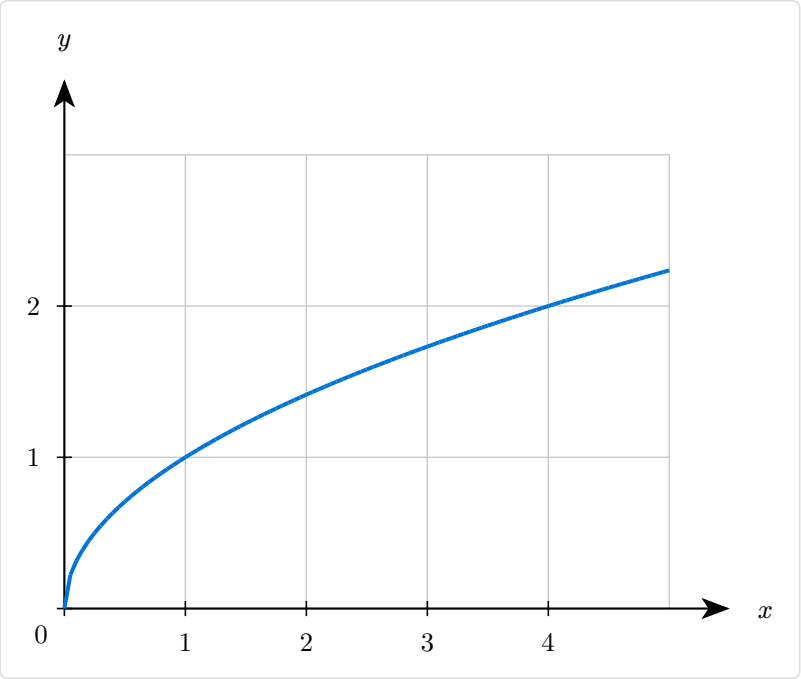
2.3. Function Domain

Specify a custom domain for functions:

Code

```
#plot(
  width: 8, height: 6,
  xmin: 0, xmax: 5, ymin: 0, ymax: 3,
  xlabel: $x$, ylabel: $y$,
  axis-x-pos: "bottom", axis-y-pos: "left",
  show-grid: "major",
  (fn: x => calc.sqrt(x), domain: (0, 5), stroke: blue + 1.5pt),
)
```

Preview



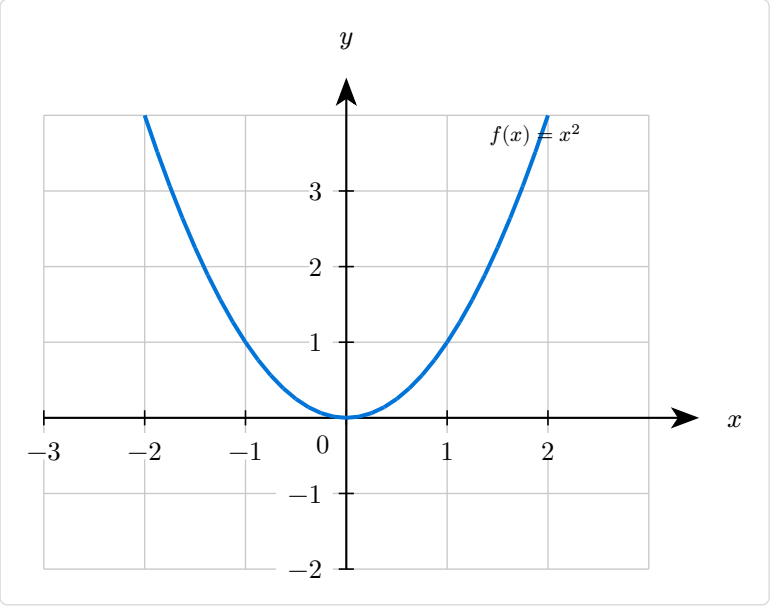
2.4. Function Labels

Add labels to your functions:

Code

```
#plot(
  width: 8, height: 6,
  xmin: -3, xmax: 3, ymin: -2, ymax: 4,
  xlabel: $x$, ylabel: $y$,
  show-grid: "major",
  (
    fn: x => x * x,
    stroke: blue + 1.5pt,
    label: $f(x) = x^2$,
    label-side: "above",
    label-pos: 0.75,
  ),
)
```

Preview



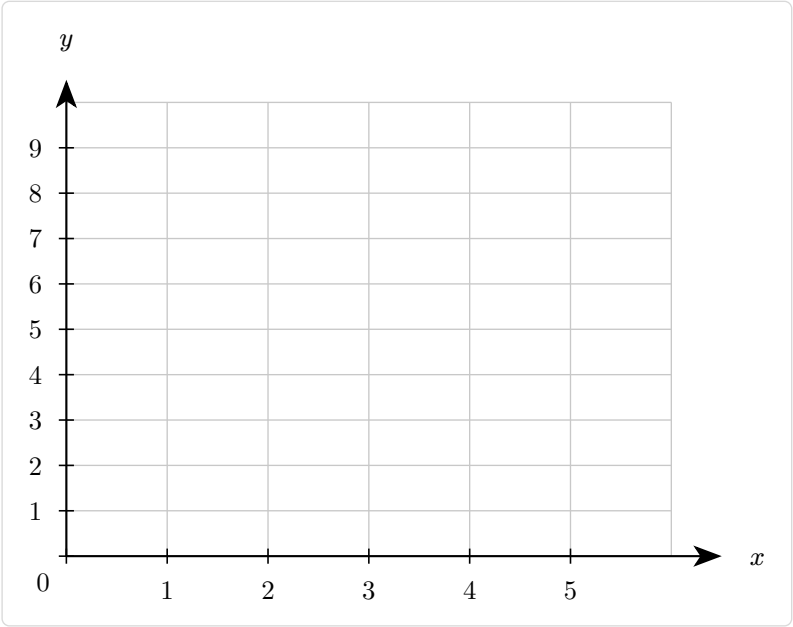
2.5. Data Points and Scatter Plots

Plot discrete data points:

Code

```
#plot(
  width: 8, height: 6,
  xmin: 0, xmax: 6, ymin: 0, ymax: 10,
  xlabel: $x$, ylabel: $y$,
  axis-x-pos: "bottom", axis-y-pos: "left",
  show-grid: "major",
  (
    data: ((1, 2), (2, 4), (3, 5), (4, 7), (5, 9)),
    mark: "o",
    mark-size: 0.15,
    stroke: blue + 1pt,
  ),
)
```

Preview



3. Grid Options

3.1. Grid Modes

Control grid display with `show-grid`:

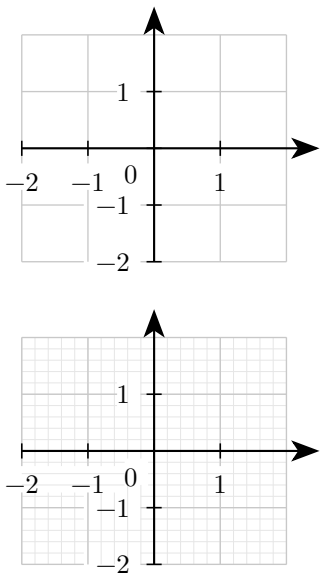
Code

```
// Major grid only
#plot(
  width: 5, height: 4,
  xmin: -2, xmax: 2, ymin: -2, ymax: 2,
  show-grid: "major",
)

// Minor grid only
#plot(
  width: 5, height: 4,
  xmin: -2, xmax: 2, ymin: -2, ymax: 2,
  show-grid: "minor",
)

// Both grids
#plot(
  width: 5, height: 4,
  xmin: -2, xmax: 2, ymin: -2, ymax: 2,
  show-grid: "both",
)
```

Preview



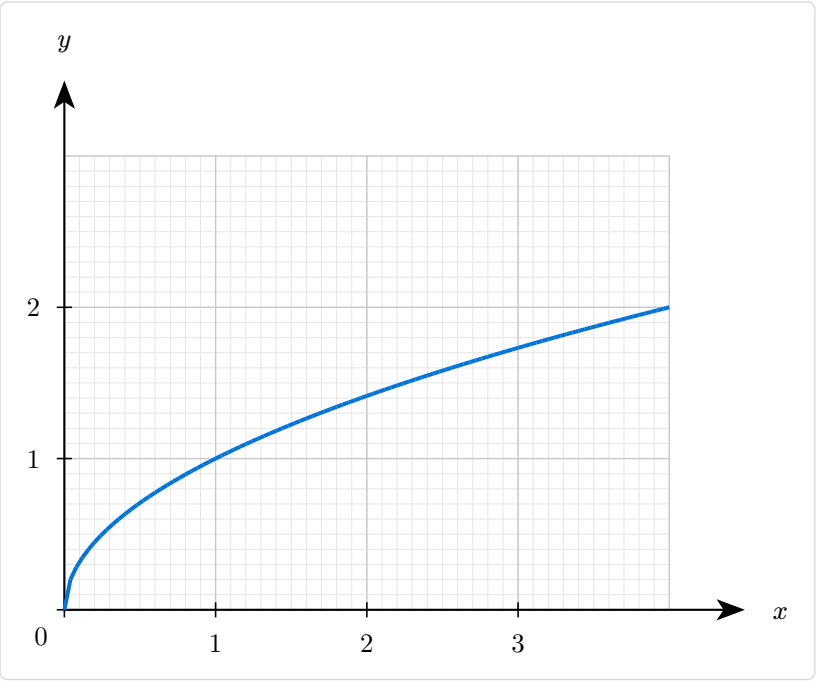
3.2. Minor Grid Subdivisions

Control the number of subdivisions with `minor-grid-step`:

Code

```
#plot(
  width: 8, height: 6,
  xmin: 0, xmax: 4, ymin: 0, ymax: 3,
  xlabel: $x$, ylabel: $y$,
  axis-x-pos: "bottom", axis-y-pos: "left",
  show-grid: "both",
  minor-grid-step: 10,  // 10 subdivisions per unit
  (fn: x => calc.sqrt(x), domain: (0, 4), stroke: blue + 1.5pt),
)
```

Preview



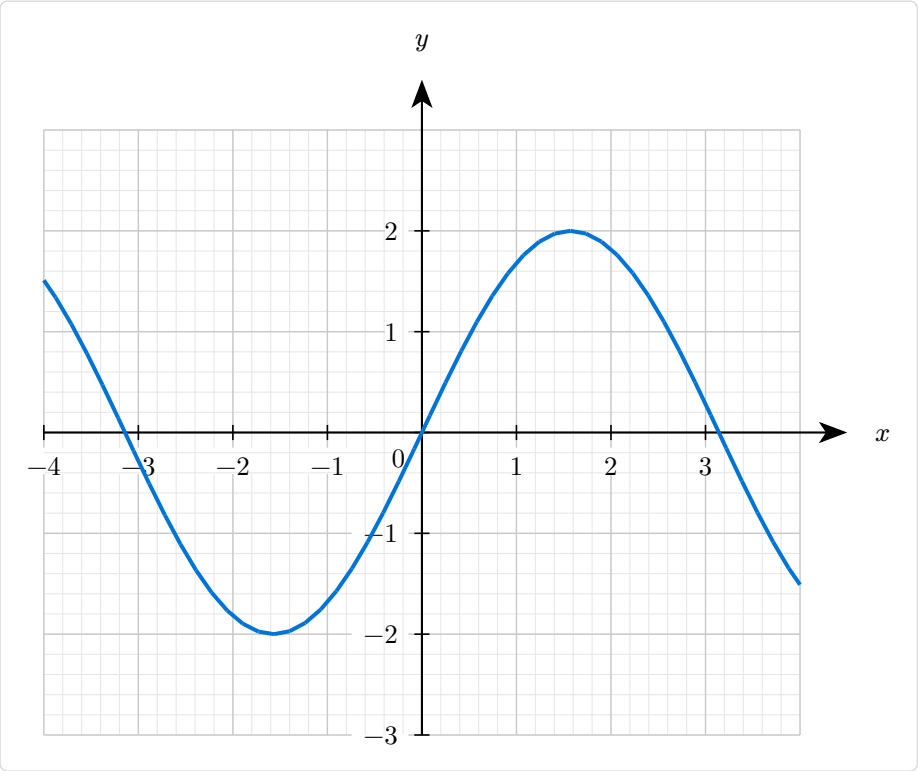
3.3. Grid Label Break

The `grid-label-break` option (enabled by default) draws grid lines with gaps around tick labels, creating an elegant break effect. Unlike a white-box approach, this works on any background color:

Code

```
#plot(
  width: 10, height: 8,
  xmin: -4, xmax: 4, ymin: -3, ymax: 3,
  xlabel: $x$, ylabel: $y$,
  show-grid: "both",
  minor-grid-step: 5,
  grid-label-break: true,  // Default
  (fn: x => calc.sin(x) * 2, stroke: blue + 1.5pt),
)
```

Preview



4. Axis Configuration

4.1. Axis Position

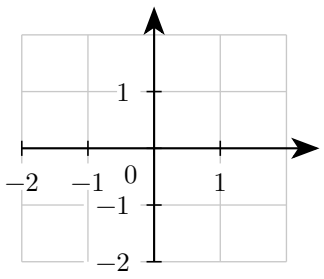
Position axes at origin (default), bottom/left, or custom values:

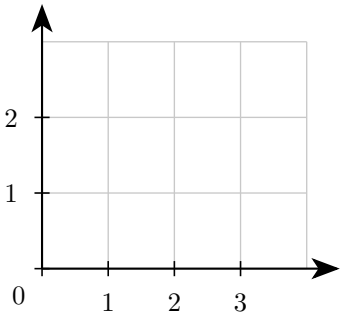
Code

```
// Through origin (default)
#plot(
  width: 5, height: 4,
  xmin: -2, xmax: 2, ymin: -2, ymax: 2,
  show-grid: "major",
)

// Bottom and left
#plot(
  width: 5, height: 4,
  xmin: 0, xmax: 4, ymin: 0, ymax: 3,
  axis-x-pos: "bottom",
  axis-y-pos: "left",
  show-grid: "major",
)
```

Preview





4.2. Axis Extension

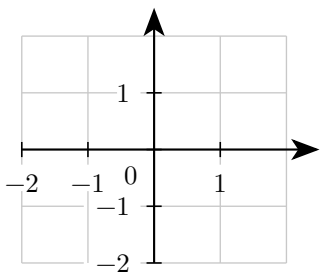
By default, axes extend 0.5 units beyond the grid on the arrow side. Customize with `axis-x-extend` and `axis-y-extend`:

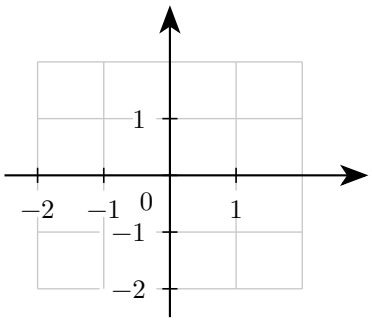
Code

```
// Default extension (0, 0.5)
#plot(
  width: 5, height: 4,
  xmin: -2, xmax: 2, ymin: -2, ymax: 2,
  show-grid: "major",
)

// Custom extension
#plot(
  width: 5, height: 4,
  xmin: -2, xmax: 2, ymin: -2, ymax: 2,
  axis-x-extend: (0.5, 1),
  axis-y-extend: (0.5, 1),
  show-grid: "major",
)
```

Preview





5. Tick Configuration

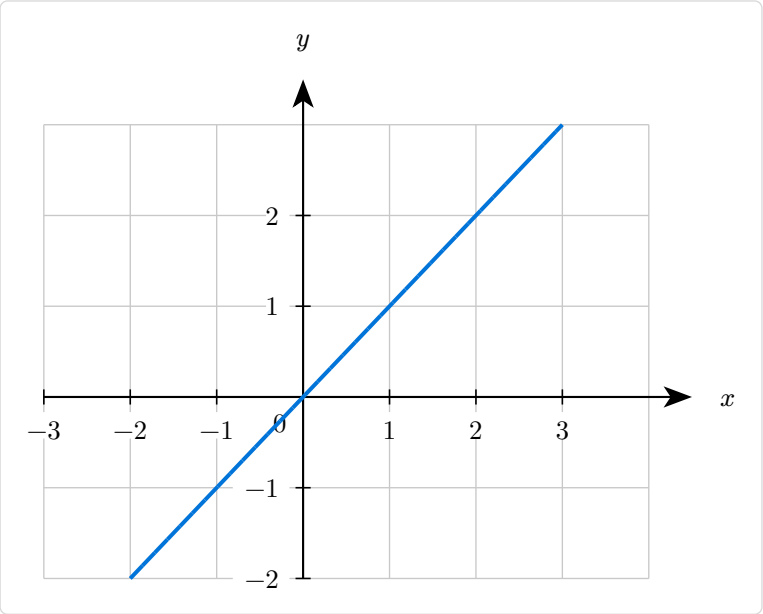
5.1. Default Integer Ticks

By default, ticks are placed at every integer (step = 1):

Code

```
#plot(
  width: 8, height: 6,
  xmin: -3, xmax: 4, ymin: -2, ymax: 3,
  xlabel: $x$, ylabel: $y$,
  show-grid: "major",
  (fn: x => x, stroke: blue + 1.5pt),
)
```

Preview



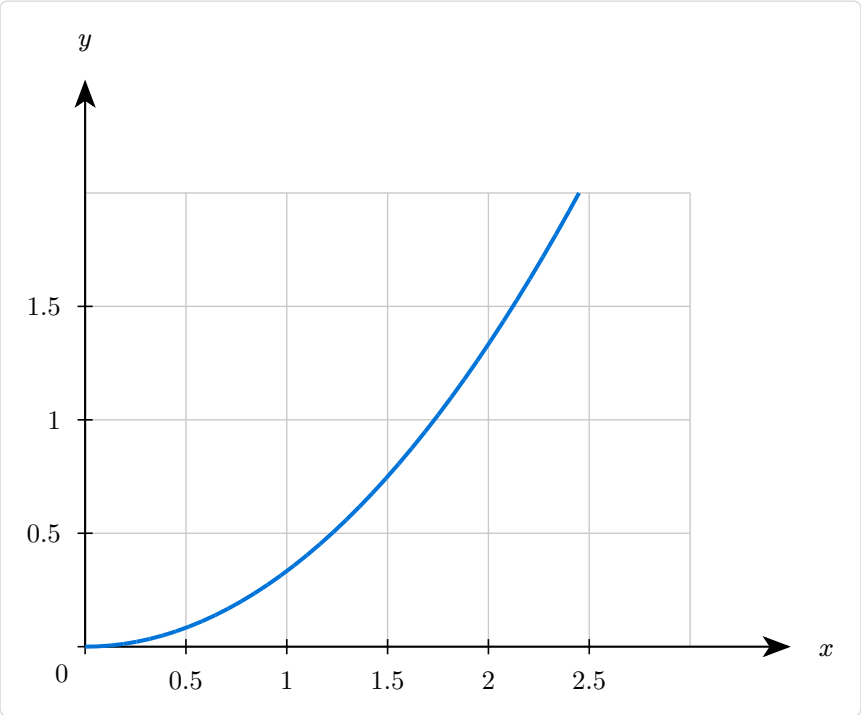
5.2. Custom Tick Step

Change tick spacing with `xtick-step` and `ytick-step`:

Code

```
#plot(
  width: 8, height: 6,
  xmin: 0, xmax: 3, ymin: 0, ymax: 2,
  xlabel: $x$, ylabel: $y$,
  axis-x-pos: "bottom", axis-y-pos: "left",
  xtick-step: 0.5,
  ytick-step: 0.5,
  show-grid: "major",
  (fn: x => x * x / 3, stroke: blue + 1.5pt),
)
```

Preview



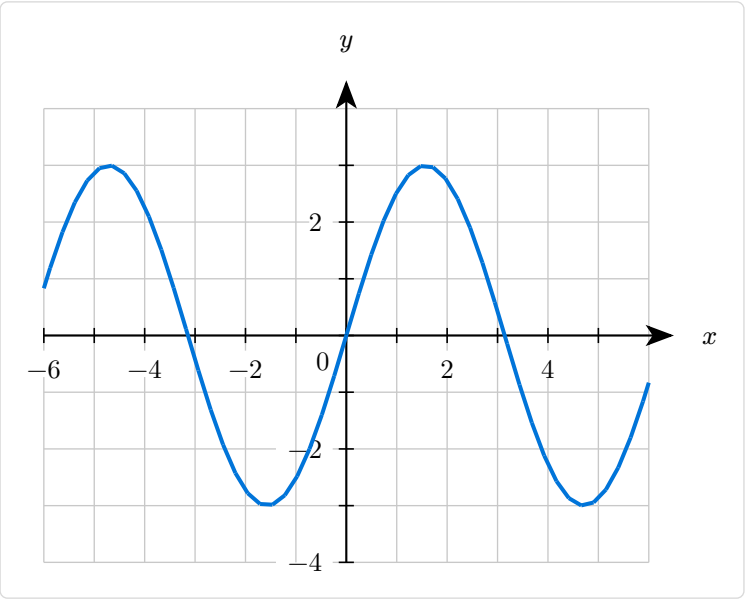
5.3. Tick Label Step

Show labels only at every N-th tick with `xtick-label-step` and `ytick-label-step`:

Code

```
#plot(
  width: 8, height: 6,
  xmin: -6, xmax: 6, ymin: -4, ymax: 4,
  xlabel: $x$, ylabel: $y$,
  xtick-label-step: 2,  // Labels at -6, -4, -2, 2, 4, 6
  ytick-label-step: 2,  // Labels at -4, -2, 2, 4
  show-grid: "major",
  (fn: x => calc.sin(x) * 3, stroke: blue + 1.5pt),
)
```

Preview



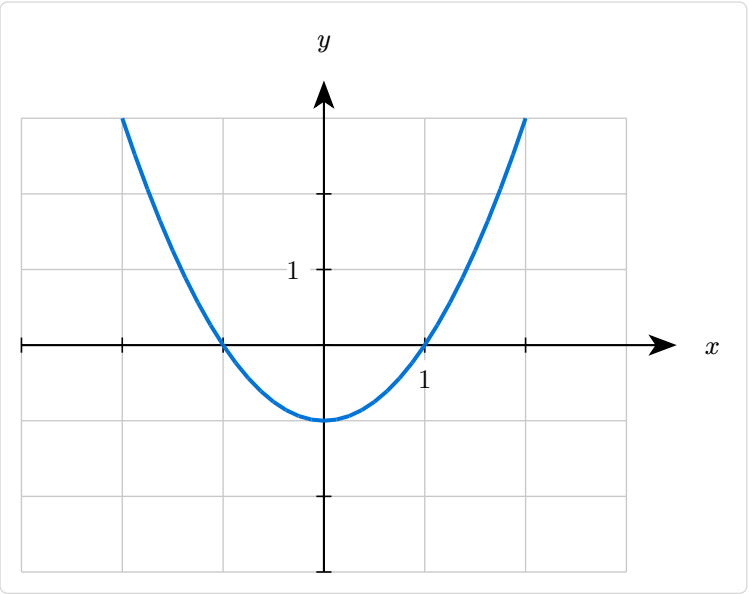
5.4. Unit Label Only

Show only “1” on each axis for a minimal style:

Code

```
#plot(
  width: 8, height: 6,
  xmin: -3, xmax: 3, ymin: -3, ymax: 3,
  xlabel: $x$, ylabel: $y$,
  unit-label-only: true,
  show-origin: false,
  show-grid: "major",
  (fn: x => x * x - 1, stroke: blue + 1.5pt),
)
```

Preview



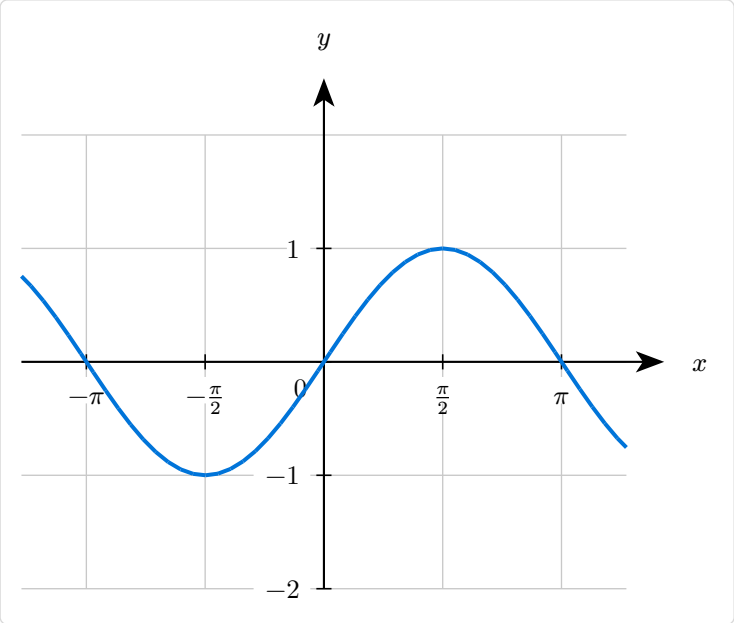
5.5. Custom Tick Positions

Specify exact tick positions with `xtick` and `ytick`:

Code

```
#plot(
  width: 8, height: 6,
  xmin: -4, xmax: 4, ymin: -2, ymax: 2,
  xlabel: $x$, ylabel: $y$,
  xtick: (-calc.pi, -calc.pi/2, 0, calc.pi/2, calc.pi),
  xtick-labels: ($-pi$, $-pi/2$, $0$, $pi/2$, $pi$),
  show-grid: "major",
  (fn: x => calc.sin(x), stroke: blue + 1.5pt),
)
```

Preview



5.6. Hide Origin Label

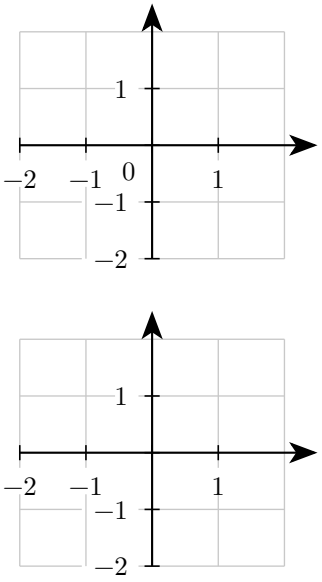
Control the “0” label at the origin:

Code

```
// With origin (default)
#plot(
  width: 5, height: 4,
  xmin: -2, xmax: 2, ymin: -2, ymax: 2,
  show-origin: true,
  show-grid: "major",
)

// Without origin
#plot(
  width: 5, height: 4,
  xmin: -2, xmax: 2, ymin: -2, ymax: 2,
  show-origin: false,
  show-grid: "major",
)
```

Preview



6. Markers

6.1. Available Marker Types

The following markers are available:

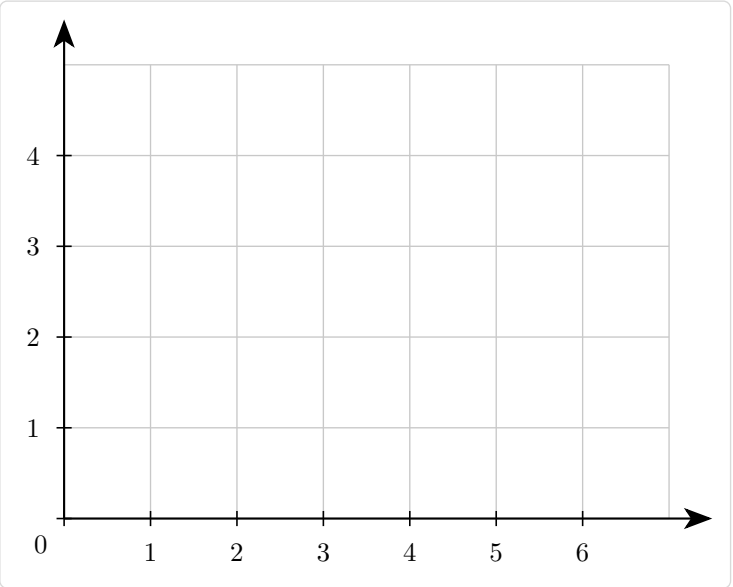
Marker	Description
`"o"`	Circle (outline)
`"*"`	Circle (filled)
`"square"` / `"s"`	Square (outline)
`"square*"`	Square (filled)
`"triangle"` / `"^"`	Triangle up (outline)
`"triangle*"`	Triangle up (filled)
`"diamond"` / `"d"`	Diamond (outline)
`"diamond*"`	Diamond (filled)
`"star"`	Star (outline)
`"star*"`	Star (filled)
`"+"`	Plus sign
`"x"`	Cross
`"|"`	Vertical bar
`"-"`	Horizontal bar

6.2. Using Markers

Code

```
#plot(
  width: 8, height: 6,
  xmin: 0, xmax: 7, ymin: 0, ymax: 5,
  axis-x-pos: "bottom", axis-y-pos: "left",
  show-grid: "major",
  (data: ((1, 1), (2, 2), (3, 2.5)), mark: "o", stroke: blue),
  (data: ((1, 2), (2, 3), (3, 3.5)), mark: "square*", stroke: red),
  (data: ((1, 3), (2, 4), (3, 4.2)), mark: "triangle", stroke: green),
)
```

Preview



7. Convenience Functions

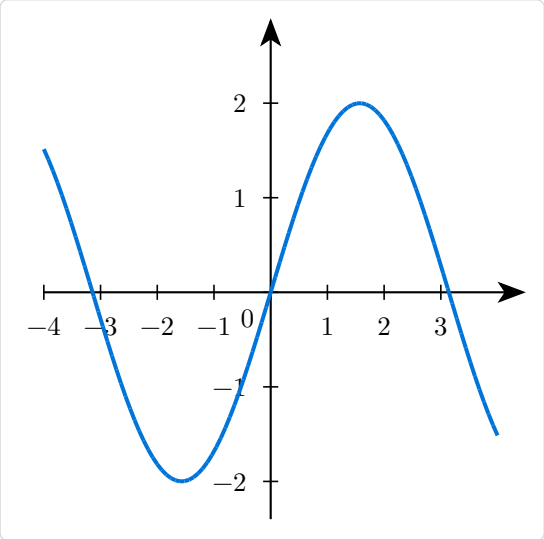
7.1. `plot-fn` - Quick Function Plot

Plot a single function with automatic y-scaling:

Code

```
#plot-fn(
  x => calc.sin(x) * 2.0,
  domain: (-4, 4),
  stroke: blue + 1.5pt,
)
```

Preview



7.2. `scatter` - Scatter Plot Helper

Create scatter plot specifications:

```
#let my-data = scatter(
  ((1, 2), (2, 4), (3, 5)),
  mark: "o",
  stroke: blue,
)
#plot(xmin: 0, xmax: 4, ymin: 0, ymax: 6, my-data)
```

7.3. `line-plot` - Line Plot Helper

Create connected line plots:

```
#let my-line = line-plot(
  ((1, 2), (2, 4), (3, 5)),
  stroke: blue + 1pt,
  mark: "o",
)
#plot(xmin: 0, xmax: 4, ymin: 0, ymax: 6, my-line)
```

7.4. `func-plot` - Function Plot Helper

Create function plot specifications:

```
#let my-func = func-plot(
  x => calc.sin(x),
```

```
  stroke: blue + 1.5pt,
  label: $sin(x)$,
)
#plot(xmin: -4, xmax: 4, ymin: -1.5, ymax: 1.5, my-func)
```

8. Global Configuration

8.1. Setting Defaults

Use `set-plot-defaults` to configure defaults for all subsequent plots:

```
#set-plot-defaults(
  width: 8,
  height: 6,
  show-grid: "both",
  minor-grid-step: 5,
)

// All plots will now use these defaults
#plot(xmin: -3, xmax: 3, ymin: -2, ymax: 2, ...)
```

8.2. Resetting Defaults

Reset to original defaults:

```
#reset-plot-defaults()
```

9. Styling

9.1. Custom Styles

Override default styles with the `style` parameter:

```
#plot(
  xmin: -3, xmax: 3, ymin: -2, ymax: 2,
  style: (
    axis: (stroke: black + 1pt, arrow: "stealth"),
    grid: (
      major: (stroke: luma(180) + 0.6pt),
      minor: (stroke: luma(220) + 0.3pt),
    ),
    ticks: (
      length: 0.12,
      stroke: black + 0.6pt,
      label-size: 0.7em,
    ),
  ),
  ...
)
```

9.2. Default Style Values

Property	Default	Description
`axis.stroke`	`black + 0.8pt`	Axis line style
`axis.arrow`	`"stealth"`	Arrow head style
`grid.major.stroke`	`luma(200) + 0.5pt`	Major grid line style
`grid.minor.stroke`	`luma(230) + 0.3pt`	Minor grid line style
`ticks.length`	`0.1`	Tick mark length (cm)
`ticks.stroke`	`black + 0.6pt`	Tick mark style
`ticks.label-size`	`0.65em`	Tick label font size
`ticks.label-offset`	`0.15`	Distance from tick to label
`plot.stroke`	`blue + 1.2pt`	Default function stroke
`plot.samples`	`100`	Default sample count
`marker.size`	`0.12`	Default marker size
`labels.size`	`0.8em`	Axis label font size

10. Parameter Reference

10.1. `plot` Function

Dimensions and Bounds:

Parameter	Type	Default	Description
`width`	float	6	Plot width in cm
`height`	float	6	Plot height in cm
`scale`	float	1	Scale factor for entire plot
`xmin`	float	auto	Minimum x value
`xmax`	float	auto	Maximum x value
`ymin`	float	auto	Minimum y value
`ymax`	float	auto	Maximum y value

Axis Configuration:

Parameter	Type	Default	Description
`xlabel`	content	none	X-axis label
`ylabel`	content	none	Y-axis label
`xlabel-pos`	string/array	“end”	“end”, “center”, or (x, y)
`ylabel-pos`	string/array	“end”	“end”, “center”, or (x, y)
`xlabel-anchor`	string	“west”	Anchor for x label
`ylabel-anchor`	string	“south”	Anchor for y label
`xlabel-offset`	array	(0.3, 0)	X label offset (cm)
`ylabel-offset`	array	(0, 0.3)	Y label offset (cm)
`axis-x-pos`	string/float	0	“bottom”, “center”, or y-value
`axis-y-pos`	string/float	0	“left”, “center”, or x-value
`axis-x-extend`	float/array	(0, 0.5)	X-axis extension (left, right)
`axis-y-extend`	float/array	(0, 0.5)	Y-axis extension (bottom, top)

Tick Configuration:

Parameter	Type	Default	Description
`xtick`	array/none	auto	Custom x tick positions
`ytick`	array/none	auto	Custom y tick positions
`xtick-step`	float	1	X tick spacing
`ytick-step`	float	1	Y tick spacing
`xtick-labels`	array/none	auto	Custom x tick labels
`ytick-labels`	array/none	auto	Custom y tick labels
`xtick-label-step`	int	1	Show x label every N ticks
`ytick-label-step`	int	1	Show y label every N ticks

`show-origin`	bool	true	Show “0” at origin
`unit-label-only`	bool	false	Show only “1” on axes
`tick-label-size`	length	0.65em	Tick label font size
`axis-label-size`	length	0.8em	Axis label font size

Grid Configuration:

Parameter	Type	Default	Description
`show-grid`	bool/string	false	true, false, “major”, “minor”, “both”
`minor-grid-step`	int	5	Subdivisions per major tick
`grid-label-break`	bool	true	Break grid lines around labels

Styling:

Parameter	Type	Default	Description
`style`	dictionary	none	Style overrides

10.2. Function/Data Specification

Each plot item is a dictionary with these fields:

Field	Type	Description
`fn`	function	Function to plot: `x => y`
`data`	array	Data points: `((x1, y1), (x2, y2), ...)`
`domain`	array	Function domain: `(xmin, xmax)`
`samples`	int	Number of samples for function
`stroke`	stroke	Line style
`mark`	string	Marker type
`mark-size`	float	Marker size in cm
`mark-fill`	color	Marker fill color
`label`	content	Label text
`label-pos`	string	“above”, “below”, “left”, “right”
`label-at`	float/string	x-position or “start”/“end”/“center”
`label-anchor`	string	Text anchor point

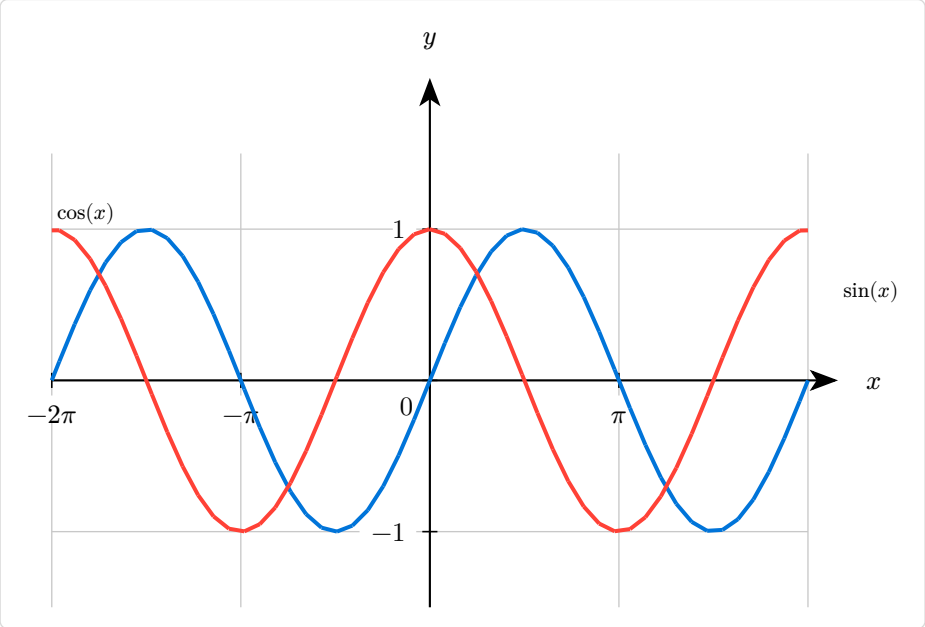
11. Complete Examples

11.1. Trigonometric Functions

Code

```
#plot(
  width: 10, height: 6,
  xmin: -2 * calc.pi, xmax: 2 * calc.pi,
  ymin: -1.5, ymax: 1.5,
  xlabel: $x$, ylabel: $y$,
  xtick: (-2*calc.pi, -calc.pi, 0, calc.pi, 2*calc.pi),
  xtick-labels: ($-2pi$, $-pi$, $0$, $pi$, $2pi$),
  show-grid: "major",
  (fn: x => calc.sin(x), stroke: blue + 1.5pt, label: $sin(x)$, label-pos: 1),
  (fn: x => calc.cos(x), stroke: red + 1.5pt, label: $cos(x)$, label-pos: 0),
)
```

Preview

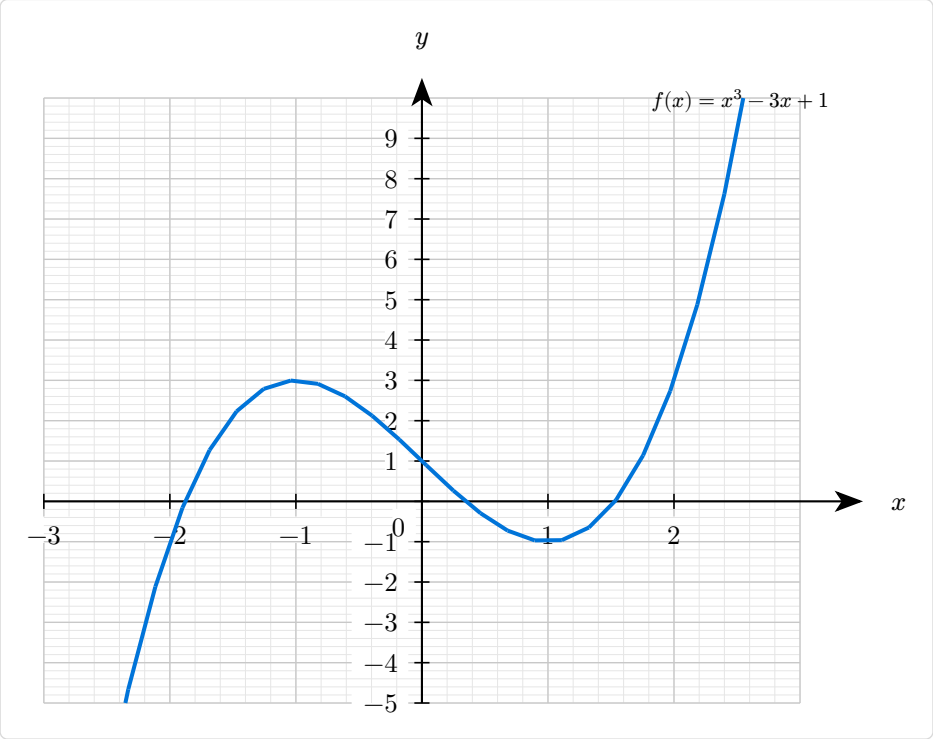


11.2. Polynomial with Fine Grid

Code

```
#plot(
  width: 10, height: 8,
  xmin: -3, xmax: 3, ymin: -5, ymax: 10,
  xlabel: $x$, ylabel: $y$,
  show-grid: "both",
  minor-grid-step: 5,
  (
    fn: x => x * x * x - 3 * x + 1,
    stroke: blue + 1.5pt,
    label: $f(x) = x^3 - 3x + 1$,
    label-side: "above",
    label-pos: 0.85,
  ),
)
```

Preview

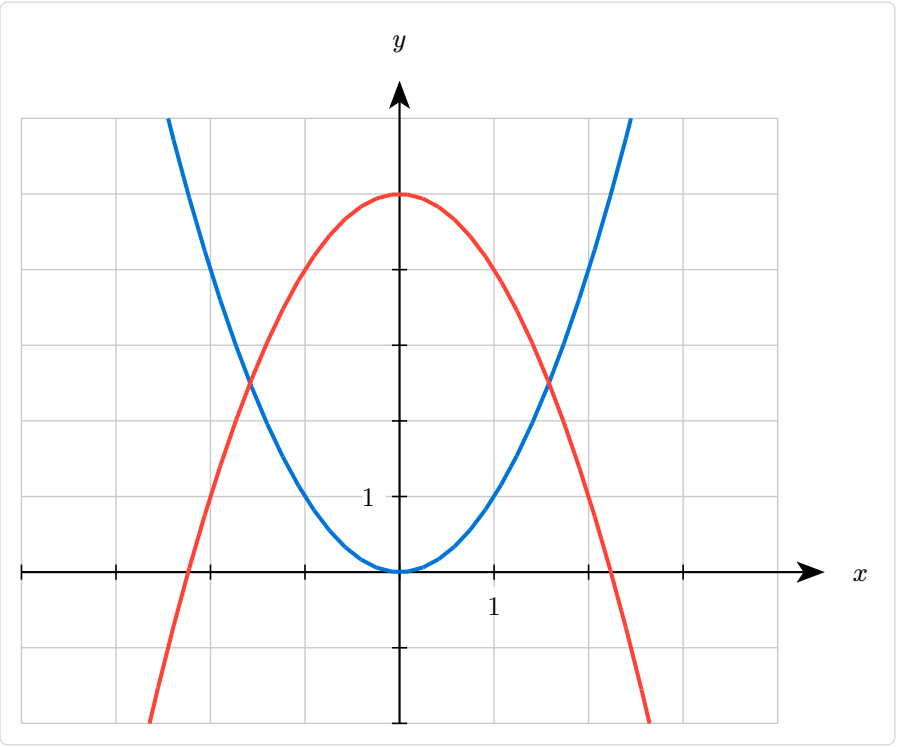


11.3. Minimal Style Plot

Code

```
#plot(
  width: 10, height: 8,
  xmin: -4, xmax: 4, ymin: -2, ymax: 6,
  xlabel: $x$, ylabel: $y$,
  show-grid: "major",
  unit-label-only: true,
  show-origin: false,
  (fn: x => x * x, stroke: blue + 1.5pt),
  (fn: x => -x * x + 5, stroke: red + 1.5pt),
)
```

Preview



11.4. Data with Trend Line

Code

```
#plot(
  width: 10, height: 7,
  xmin: 0, xmax: 6, ymin: 0, ymax: 12,
  xlabel: "Time (s)", ylabel: "Distance (m)",
  axis-x-pos: "bottom", axis-y-pos: "left",
  show-grid: "both",
  minor-grid-step: 5,
  (fn: x => 2 * x, stroke: gray + 1pt, domain: (0, 6)),  // Trend line
  (
    data: ((0.5, 1.2), (1, 2.3), (2, 3.8), (3, 6.2), (4, 7.9), (5, 10.1)),
    mark: "o",
    mark-size: 0.12,
    stroke: none,
  ),
)
```

Preview