LeanPlot

LeanPlot is a **tiny plotting wrapper for Lean 4's `ProofWidgets4` ecosystem**. It lets you write Lean code that produces interactive charts—rendered directly in VS Code's infoview—using the rich Recharts React library.

The goal is to grow a *grammar-of-graphics* style API over time. For now we provide convenience helpers around the most common use-cases: sampling Lean functions and visualising the result as line charts.

Features (0.2.x)

Legacy features (0.2.x – superseded)

- **Tier-0 zero-config helpers** `LeanPlot.API.lineChart` and `scatterChart` – go from a Lean function *or* an array of points to an interactive plot with **one line of Lean**.
- **Composable graphics algebra** – build plots from smaller pieces and overlay them with the ordinary `+` operator:

  ```
  import LeanPlot.Algebra; open LeanPlot.Algebra

  #plot (line "y"  (fun x   x) +
         line "y²" (fun x   x*x))
  ```

- `sample` / `sampleMany` – lower-level helpers to uniformly sample functions on an interval (works for any codomain that has a `[ToFloat]` instance).
- `mkLineChart` / `mkScatterChart` – escape hatches that let you customise every Recharts prop once you outgrow Tier-0.
- Ready-to-run demos under `LeanPlot/Demos` (linear, quadratic, cubic, overlay, area, bar).

Features (0.3.x-alpha)

- **Tier-0 zero-config helpers** `LeanPlot.API.lineChart` and `scatterChart` – go from a Lean function *or* an array of points to an interactive plot with **one line of Lean**.
- **Composable graphics algebra** – build plots from smaller pieces and overlay them with the ordinary `+` operator *or* the explicit `PlotSpec.stack` helper:

  ```
  import LeanPlot.Algebra; open LeanPlot.Algebra

  #plot (line "y"  (fun x   x) +
         line "y²" (fun x   x*x))
  ```

- **Layered specification language** – low-level `PlotSpec` API for fine-grained control; new `overlay/stack` combinators let you combine two `PlotSpecs` just like higher-level plots.
- `sample` / `sampleMany` – lower-level helpers to uniformly sample functions on an interval (works for any codomain that has a `[ToFloat]` instance).
- `mkLineChart` / `mkScatterChart` – escape hatches that let you customise every Recharts prop once you outgrow Tier-0.
- Ready-to-run demos under `LeanPlot/Demos` (linear, quadratic, cubic, overlay, area, bar, **stack**).

Installation

Add LeanPlot as a dependency in your project's `lakefile.toml`:

```toml
[[require]]
name = "LeanPlot"
url = "https://github.com/alok/LeanPlot"
```

or in `lakefile.lean`:

```
require LeanPlot from git
  "https://github.com/alok/LeanPlot" @ "main"
```

Then run:

```
lake update
lake build
```

Make sure you have node/npm installed—the ProofWidgets build will take care of JS bundling automatically.

Quick start

Open a `.lean` file in VS Code with the infoview visible and paste:

```
import LeanPlot.Algebra

open LeanPlot.Algebra

#plot (
  line "y"  (fun x : Float   x)
  + -- '+' is an alias for `PlotSpec.stack`, overlaying the two plots
  line "y²" (fun x   x*x)
)
```

You should see two series rendered in a single interactive chart.

Demo gallery

See `Gallery.md` for the roadmap of examples we plan to support. The following are already available:

- `LeanPlot.Demos.LinearDemo` – `y = x`
- `LeanPlot.Demos.QuadraticDemo` – `y = x²`
- `LeanPlot.Demos.CubicDemo` – `y = x³`
- `LeanPlot.Demos.OverlayDemo` – overlay of `y = x` and `y = x²`
- `LeanPlot.Demos.StackDemo` – stacking/overlay via `+` and `stack` helpers

Run them by putting your cursor over the `#html` command in each file.

Development

```
just build         # = lake build
just linter        # run Std.Tactic.Lint (setup WIP)
just docs          # regenerate docs (TBD)
```

Contributions welcome! Check the TODO list and open a PR or issue.

Licence

LeanPlot is released under the Apache License, Version 2.0. See `LICENSE.txt` for details.