The FlatPPL CLI

COMMAND HELP

```
flatppl <command> --help
```

WRITE AND CHECK

The input file · score.flatppl

```
flatppl_compat = "0.1"
x = elementof(reals)
mu = elementof(reals)
inputs = (x, mu)
outputs = logdensityof(Normal(mu, 1.0), x)
K = functionof(Normal(mu, 1.0))
L = likelihoodof(K, 1.0)
prior = joint(mu = Normal(0.0, 1.0))
posterior = bayesupdate(L, prior)
```

Save this listing as `score.flatppl` in an empty folder. Run the commands there. The query scores a Normal reading. The posterior uses the observed value 1.

Check or apply formatting

```
flatppl fmt --check score.flatppl
flatppl fmt score.flatppl
flatppl fmt - < score.flatppl > clean.flatppl
```

Arguments to `fmt`	Result
`--check FILE`	Exit 1 if the file needs formatting.
`FILE`	Format the file in place.
`-` or no path	Format stdin to stdout.

Check diagnostics

```
flatppl lint --deny-warnings score.flatppl
```

Promote warnings to errors for CI. Use `--allow`, `--warn`, or `--deny` with a rule name to set its level.

Find commands and flags

```
flatppl --version
flatppl --help
flatppl help stablehlo
flatppl completions zsh
```

Top-level help lists the commands in the installed build. Subcommand help lists options and defaults. The last command prints a shell completion script.

Read an exit status

`0` Command succeeded.
`1` Input, I/O or lint error, or formatting differs.
`2` Invalid command-line option.
`3` The chosen compiler cannot emit the model.

INSPECT AND CONVERT

Infer types and shapes

```
flatppl infer score.flatppl typed.flatpir
```

Each `--level` includes the preceding levels.

Level	Information added to FlatPIR
`phase`	Binding phases
`type`	Structural types and literal dimensions
`valueset`	Strongest known value set
`normalization`	Mass classes for measures and kernels
`shape` (default)	Resolved fixed-phase dimensions

Trace parameter dependence

```
flatppl analyze --data-input x score.flatppl
```

Mark `x` as data. The report identifies `mu` as the sole parameter component and shows which inputs each log-density term uses.

Inspect the inferred value sets

```
flatppl infer --level valueset --no-header \
  score.flatppl sets.flatpir
```

Read `%meta` annotations in the output. The selected level leaves later fields unresolved.

Convert or typeset a model

```
flatppl convert score.flatppl score.flatpir
flatppl convert typed.flatpir view.flatppl
flatppl convert score.flatppl score.typ
```

File extensions select the formats. Use `.flatpir.json` for JSON or `.tex`, `.typ`, `.md`, or `.html` for mathematical notation. `--no-header` omits the generated-file banner.

Fetch module dependencies

```
flatppl prepare score.flatppl
```

The printed model uses built-ins. For models with remote imports, `prepare` fetches and caches them after approval. Set `FLATPPL_CACHEDIR` to choose the cache for offline use.

Import a model

```
flatppl convert --from pyhf ws.json fit.flatppl
flatppl convert --from hs3 hs.json h.flatppl
flatppl convert r.formula.json r.flatppl
```

Extract the three JSON inputs from this PDF. The pyhf and HS³ importers use the `hs3` feature. Formula specifications use `formula`.

COMPILE AND SHARE

Emit deterministic source

```
flatppl determinize --keep outputs \
  score.flatppl -o pure.flatppl
```

Print deterministic FlatPDL in surface syntax. `--keep outputs` retains the selected binding and its dependencies. Use `--emit flatpir` for S-expressions.

Emit StableHLO

```
flatppl stablehlo --dtype f64 \
  score.flatppl -o score.mlir
```

The output is textual StableHLO. Sample mode requires a model with a sampling signature.

Option	Default	Alternative
`--mode`	`logdensity`	`sample`
`--dtype`	`f32`	`f64`
`--target`	`cpu`	`gpu`

Export a Stan posterior

```
flatppl stan --target posterior \
  score.flatppl -o mean.stan \
  --data mean.json
```

The `stan` feature writes the model and its data. Stan sampling statements omit additive log-density terms independent of the parameters.

Map Stan names to model fields

```
flatppl stan --target posterior \
  --manifest names.json score.flatppl \
  -o mean.stan
```

The JSON manifest maps Stan parameters to FlatPPL fields. Here `mu` has the same name in both.

Pack a model

```
flatppl pack score.flatppl score.flatppl.zip
```

Prepare remote imports first. The new archive includes the entry model, its imports, and referenced data.

Open a portable model

```
flatppl infer score.flatppl.zip packed.flatpir
flatppl unpack score.flatppl.zip score-copy
```

Inspect a ZIP directly, or extract it into a new directory to edit its contents. The entry file is `main.flatppl`.

Jessica Cox · bcox@mpp.mpg.de