Project Proposal: Advancing BinnedDistributionFit.jl for HEP Analysis

Divyansh Goyal
{ Guru Gobind Singh Indraprastha University **}**
Mentors : **Dr. Jerry Ling { Harvard University }**

Abstract:

High Energy Physics (HEP) analyses heavily rely on binned likelihood fits. This project proposes to significantly develop and mature BinnedDistributionFit.jl, with the following goals: (1) Integrating the popular optimizer in HEP, Minuit2.jl, with the Optimization.jl interface. (2) Expanding the available likelihood functions (e.g., binned Chi-squared) and model types to cover basics from `RooFit`, such as ExtendPdf and SumPdf (3) JuliaHEP ecosystem integration such as supporting building likelihood from FHist.jl (4) Establishing comprehensive documentation and testing.
In the JuliaHEP fitting ecosystem, this package will complement the unbinned fit focused AlgebraPDF.jl.

Meta: Start - **20th May, 2025** | Time Commitment - **> 40hrs/week**
Parallel commitments - **Semester Examination (Recurring prior mid-terms have prepared us well)**

---

1. Motivation & Context:

Binned parameter estimation via likelihood or x^2 minimization is fundamental to HEP analysis, yet lacks a mature, dedicated tool in the Julia ecosystem. `BinnedDistributionFit.jl` provides a basic foundation with `ExtendPdf`/`SumOfPdfs` models and a `RooFit`-style NLL functor using `FHist.jl`, but requires significant development. This project aims to mature this package into a robust tool for binned fits, complementing the unbinned capabilities of `AlgebraPDF.jl`.

---

2. Project Goals & Technical Deliverables:

- **A. Optimizer Integration (`Minuit2.jl` via `Optimization.jl`)**:
  - **Goal:** Enable `Minuit2.jl` as a solver backend within `Optimization.jl`.
  - **Deliverable:** A functional `Optimization.jl`-compliant interface/wrapper for `Minuit2.jl`, mapping `OptimizationProblem` components (objective function, parameters, bounds, potentially gradients) to the `Minuit2.jl` API. Tested for minimizing `BinnedDistributionFit.jl` objective functions.

- **B. Expanded Objective Functions & Model Types**:
  - **Goal:** Provide core objective functions and refine foundational model representations.
  - **Deliverables:**
    - Binned x^2 objective function functor: $\Sigma\ (data_i - model_i)^2\ /\ \sigma_i^2$, correctly handling `FHist.jl` uncertainties (`sumw2`).
    - Refined `ExtendPdf`, `SumOfPdfs` implementations: Ensuring robustness, clear API, and compatibility with the new parameter system (Goal C).
    - *(Stretch Goal):* Basic histogram template models: Structure (`HistModel`?) wrapping `FHist.Hist1D` for direct use in fits (yield * template).

- **C. API Enhancement, Parameter Handling & Ecosystem Integration (`FHist.jl`)**:
  - **Goal:** Improve usability, maintainability, and interoperability.
  - **Deliverables:**
    - Refined user-facing API: Streamlined functions for model definition, likelihood/`x²` functor generation, and fit invocation via `Optimization.solve`.
    - Robust Parameter Management: Replace vector-based approach with a structure (e.g., `Parameters.jl` or `FlaggedNamedTuple`-inspired) supporting named parameters, state (fixed/floating), bounds, and clear association with model components/yields.
    - Improved `FHist.jl` Integration: Direct construction of objective functors from `FHist.Hist1D` objects; clear handling of binning and uncertainties.

- **D. Validation & Documentation**:
  - **Goal:** Ensure package correctness, stability, and user adoption.
  - **Deliverables:**
    - Comprehensive `Test.jl` suite: High coverage (>80%) of all features, including integration tests for fitting workflows.
    - CI Integration: GitHub Actions workflow executing the full test suite.
    - Formal Documentation: `Documenter.jl`-based website with installation guide, technical API reference, tutorials demonstrating NLL/`x²` fits with `Minuit2.jl`, and context within JuliaHEP.

---

4. Timeline (3 Months / 13 Weeks):

- **W1-2:** `Optimization.jl`/`Minuit2.jl` interface implementation & testing. Parameter system design.
- **W3-4:** Binned `x²` functor implementation. Parameter system core implementation. Initial unit tests.
- **W5-6:** Refactor `ExtendPdf`/`SumOfPdfs` & `RooFitNLL` for new parameter system. Integrate `x²` with `Optimization.jl`. Basic fit tests (NLL, `x²`).
- **W7-8:** Finalize & test parameter handling system (incl. state management). Refine `FHist.jl` integration points. Expand `Test.jl` suite coverage significantly.
- **W9-10:** `Documenter.jl` setup & core content writing (API stubs, intro). Develop tutorial examples. API refinement based on usage.
- **W11-12:** Complete documentation (API reference, tutorials). Final testing & code polish. *(Stretch)* Implement basic `HistModel` if feasible.
- **W13:** Prepare release candidate. Garner traction via announcements to the HEP community, over at slack, discourse and other community channels.

5. Contributor Background:

I am a sophomore at Indraprastha University, pursuing a degree in Artificial Intelligence and Machine Learning under the Computational Biology domain. As a GSoC 2024 contributor with JuliaHealth, I developed **MedEye3d.jl**, a 3D medical imaging and visualization tool, and supported the Julia ecosystem with packages for DICOM/NIFTI handling and ITK-based registration. Building on this experience and my work on **HepMC3-dev.jl**, I bring cross-domain expertise in scientific software, aiming to strengthen Julia's HEP tooling and data analysis pipelines.
