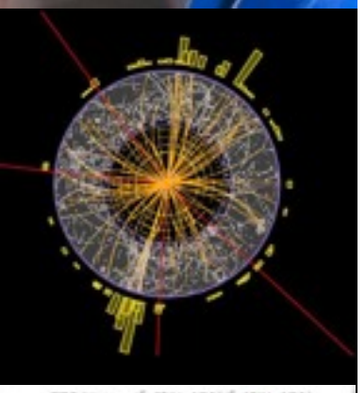
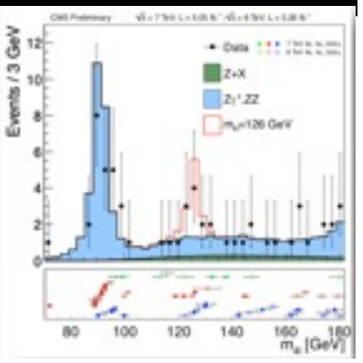
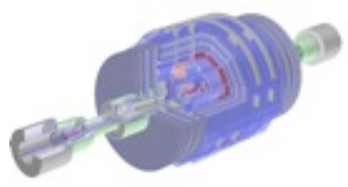






ROOT Math/Stat PoW Review

Lorenzo Moneta
PH/SFT

June 14, 2013

Overview

- What is supposed to be in ROOT 6 ?
  - Core Math Libraries
    - Random numbers
    - Fitting
    - Minimization
  - Vectorization (Vc)
  - Histograms
  - RooFit/RooStats
  - TMVA
- Code consolidation
- Available man power
- Conclusions

Core Math Developments for ROOT 6

- Random numbers
  - New vectorized version of Mersenne-Twister (TRandom3)
  - New Random number generator, Random123
    - counter based generator (stateless) ideal for parallel application
      - large amount of code (separate library ?)

- Vectorization
  - include Vc library in ROOT distribution
    - make sure it works smoothly with SMatrix/SVector and Physics Vector
  - add vectorized math functions

```
• double Gaus(double x, double m, double s);
• void Gaus(int n, const double * x, double m, double s, double * y);
• template <typename T>
  T Gaus( T x, double m, double s);
```

Fitting Improvements

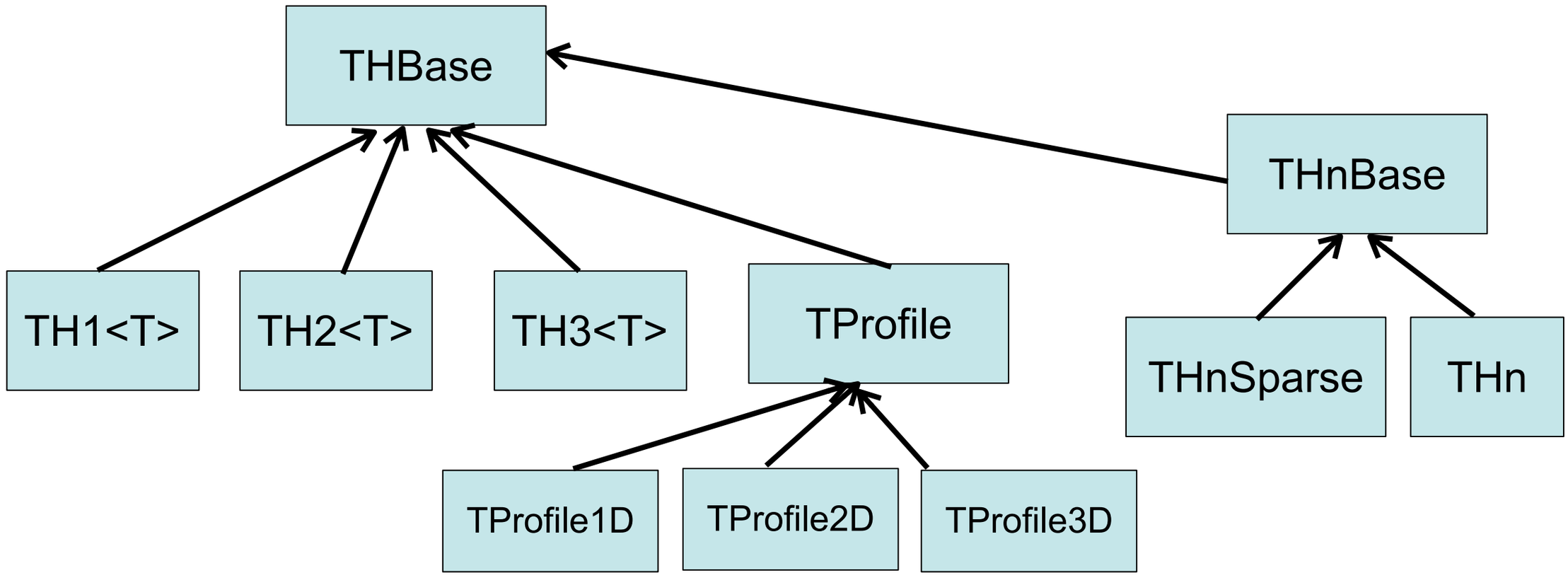
- Vectorize fit function interfaces
- Change fitting data structures from AOS to SOA
- Add a multi-thread version ?
  - not clear yet which implementation to use
    - wait for common choice in ROOT for thread support
  - ensure anyway thread safety of fitting code
- Should have PROOF support in fitting
  - need to have a PROOF plug-in
- Simplify fitter interface
  - facilitate complex fits directly in ROOT
    - building more complex models
    - building fits to combined data sets
- Better documentation

Minimization: Minuit2

- Make Minuit2 the default minimization engine when fitting (in both ROOT and RooFit)
  - reliable result
  - sometimes using less memory and faster in problems with large number of parameters
    - ATLAS Higgs combined model has ~ 1600 parameters
  - better debugging information
  - used now for complex fits at LHC (e.g Higgs results)
- Minuit2 has an OO design
  - can be improved and extended (not for ROOT 6)
    - L-BFGS algorithm
    - sparse optimization algorithms
    - non-trivial constraints
    - parallelize contours

Histogram Developments

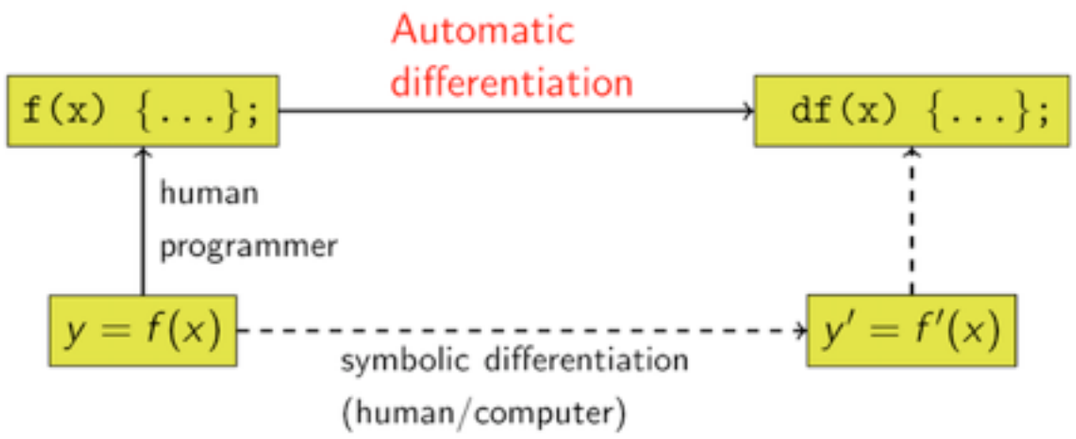
- Propose a redesign of histogram classes
  - fix the current hierarchy structure



  - ensure backward compatibility in I/O
  - THBase base class also for the THn and THnSparse
  - Fix as well TF1 structure and implement TFn

Function Improvements

- New TFormula class based on CLING/LLVM
  - JIT compilation of the expression
    - faster when evaluating dictionary functions
    - need to study possible overhead in running the JIT compilation
  - Some extensions to TFormula could then be implemented
    - easier complex functions building
    - compute derivatives using auto-differentiation
  - A GSOC student has started working on this
- Started a project on Auto-differentiation using Cling
  - with another GSOC student supervised by Vassil



RooFit

- Consolidation, no major new features
  - several performance optimizations already introduced in 5.34
- Better parallelization (through PROOF) of toy studies, event generation,...
- Vectorization of likelihood calculations
- Better organization of data representation for binned and counting data/models
  - Fit data are represented as a (STL) vector store
    - interface to create TTree on the fly from RooFit data store objects can be added
- Some code clean-up
  - remove duplications with algorithm present now in core ROOT (e.g. MathMore)

RooStats

- Bug fixes and consolidation
  - remove deprecated classes
- Extend functionality for multi-parameters problems
- Parallelization
  - support PROOF
  - More native support for parallel processing (MPI ?)
- Useful tools based on RooFit/RooStats developed in Atlas and CMS
  - some are of common utility
  - include in ROOT to better maintain and manage in the long term ?
  - need manpower for this

TMVA

- Major work is on consolidation
  - several small fixes expected for newer 5.34 versions
- Expected developments
  - increase robustness of MVA algorithm
    - less sensitivity to systematic error variations

- Critical:
  - missing man power
- TMVA is heavily used by all LHC experiments
  - code is complex, need effort to maintain and support
  - support provided now only by *H. Voss*
  - not clear what is going to happen next year

Deprecating Code

- Would like to deprecate TVirtualFitter/TFitter
  - have now new interfaces for fitting and minimization
    - TFitterMinuit (based on Minuit2) is already removed
  - no need to have a static TVirtualFitter object when fitting an histogram
- Remove also classes no longer supported (e.g. TLimit)
- Classes used very little (or not at all) should be moved in a separate optional library

- Backward compatibility
  - some improvements foreseen for ROOT 6 cannot preserve a backward compatible API
    - e.g. bit **`kCanRebin`** in histograms

Other Issues

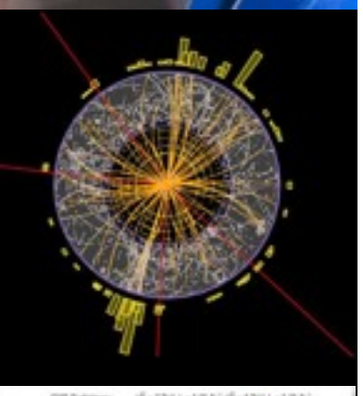
- C++11
  - would like to start using it
  - mainly internally or for transient structures (as in CMS)
- Commit to support and back port bug fixes in 5.34 for long time
- Continue to experiment and prototype new architectures
  - GPU, MIC, ARM, etc..
- Boost
  - we could make more use of some of the nice features of Boost library (Boost Math)
  - difficult to extract code from Boost
  - make it more easily available ?
    - do a wrapper library or generate the dictionary for some of the functions ?

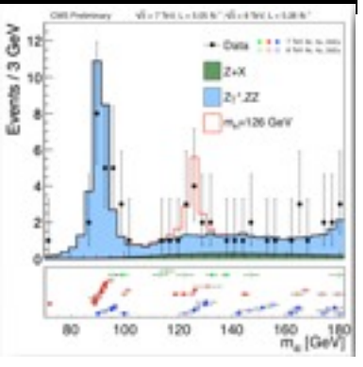
Man Power

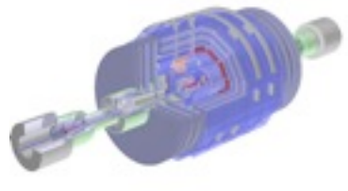
- Current contributors
  - myself (~ 80 %)
  - 2 people financed from LPCC (thanks to M. Mangano) for 4-6 months working on RooStats
  - 1 GSOC student working on TFormula
  - Had a technical student last year, not replaced so far
- Contributions from outside:
  - V. Werkerke for RooFit
  - H. Voss for TMVA (missing next year)
  - K. Cramer for RooStats/HistFactory
  - sporadic contributions from other people
    - one student (not accepted as GSOC) is working on ROOT-R interface
- Supporting the Math/Stat libraries is a large effort

Conclusions

- Several improvements expected for ROOT 6
  - vectorization
  - new histogram design
- Hope to profit soon from ROOT 6 features:
  - new TFormula class with JIT compilation
- Trying to put an effort to reduce overall code base
  - deprecate/remove/re-package not or rarely used classes
- Difficult to have all in November with current limited man power
  - supporting the current code requires time
  - some parts are critical (e.g. RooFit, TMVA....)
  - experiments rely on us







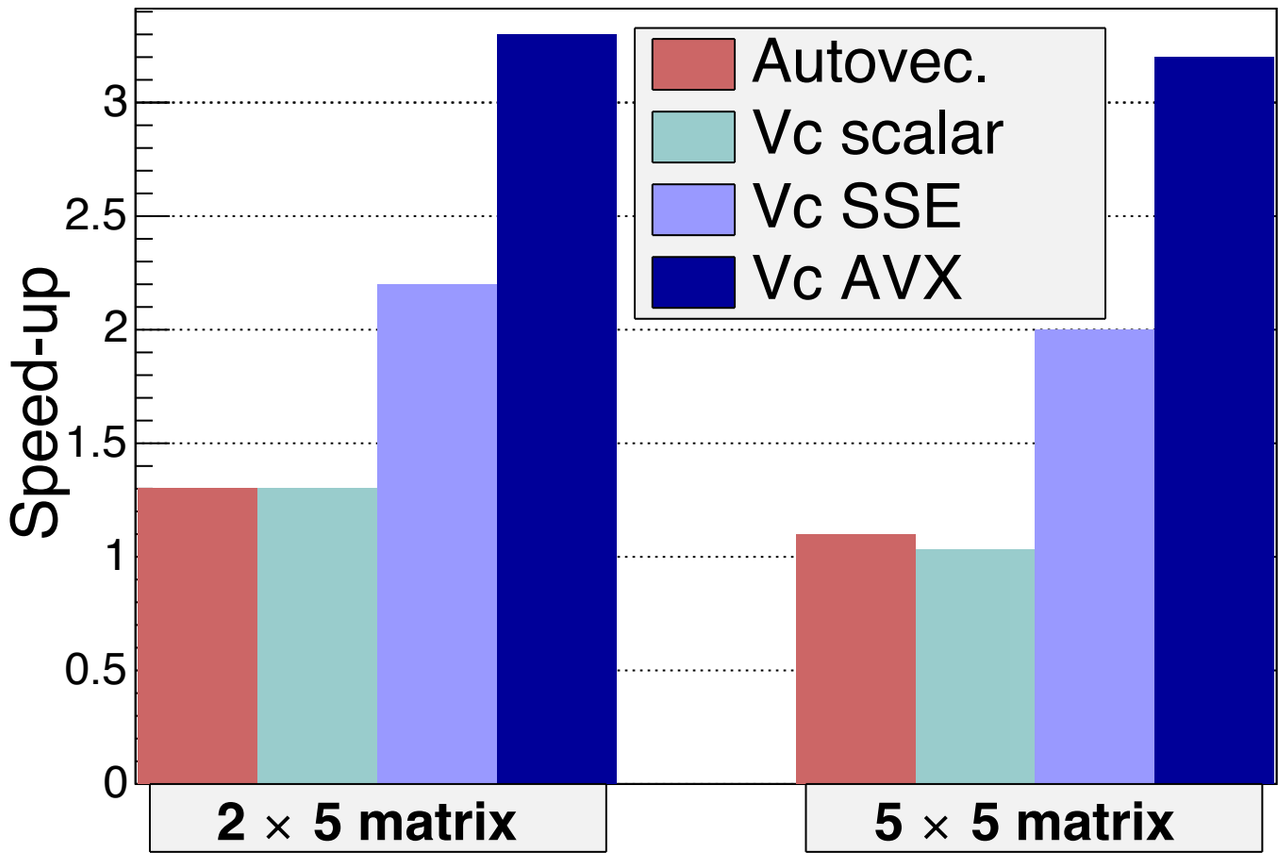
Backup Slides

Vc Library

- C++ wrapper library around intrinsic for using SIMD
  - *developed by M. Kretz (Goethe University Frankfurt) and used at GSI and BNL*
  - minimal overhead by using template classes and inline functions
- Provides vector classes (**`Vc::float_v, Vc::double_v`**) with semantics as built_in types
  - one can use **`double_v`** as a **`double`**
  - all basic operations between doubles are supported (**+,-,/,***)
  - provides also replacement for math functions (**`sqrt, exp, log, sin,...`**)
- Possible to exploit vectorization without using intrinsic and with minimal code changes
  - e.g. replace `double` → `double_v` in functions
  - easy to do in classes or functions templated on the value type
    - **`LorenzVector<PxPyPzE4D<double> >`** → **`LorenzVector<PxPyPzE4D<Vc::double_v> >`**
    - **`SMatrix<double, N1, N2 >`** → **`SMatrix<double_v, N1,N2>`**
  - vectorize loop on list of objects (vectors, matrices), reduced by **`double_v::Size`**

Vc Validation: Kalman Filter Test

- Typical operation in track reconstruction
  - very time consuming
    - inversion + several matrix-vector multiplications



- Clear advantage with Vc
  - SMatrix code can works using `double_v` as value_type
  - good boost in performance in an already performant code (5-10 times faster than CLHEP)

  - used `gcc 4.7.2` with `-mavx -O3 -fast-math`

Vectorization in Fitting

- Vectorize chi-square calculation in fitting ROOT histograms
  - *work performed by M. Borinsky (summer student 2012)*

$$\chi^2 = \sum_i \frac{(y_i - f_{a,b,\ldots}(x_i))^2}{\sigma_i^2}$$

- Required change in data set layout and in functions
  - from array of structure to structure of arrays for input data
  - vectorized function interface (TF1)

```
1 double func( double x, double* p )
{
3     return exp( - p[0] * x );
}
```

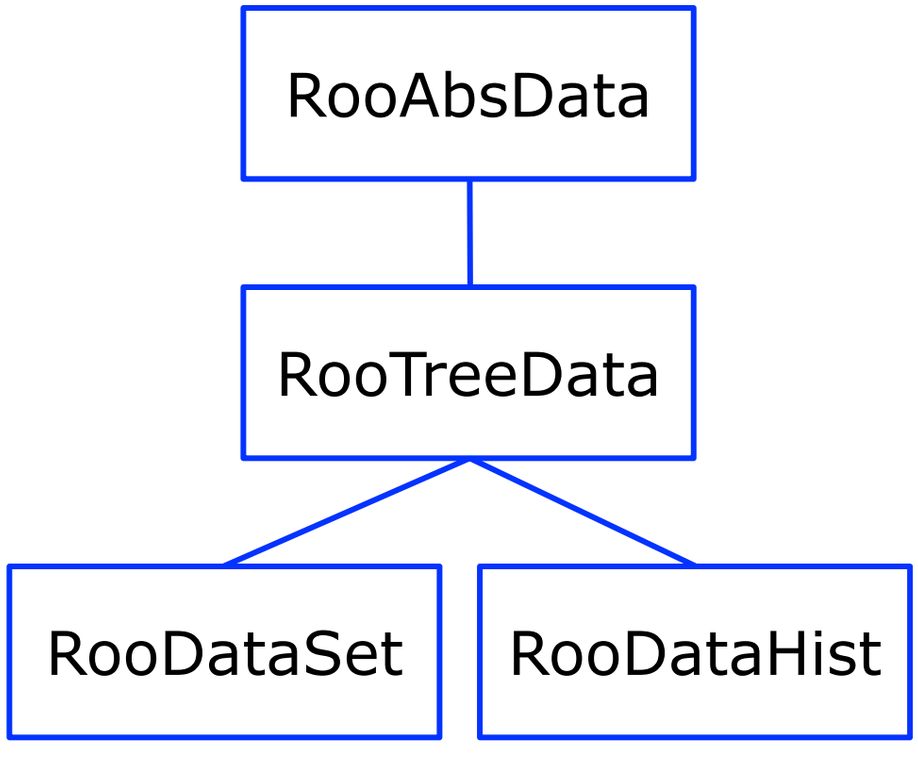
Listing 1: Old callback function for TF1

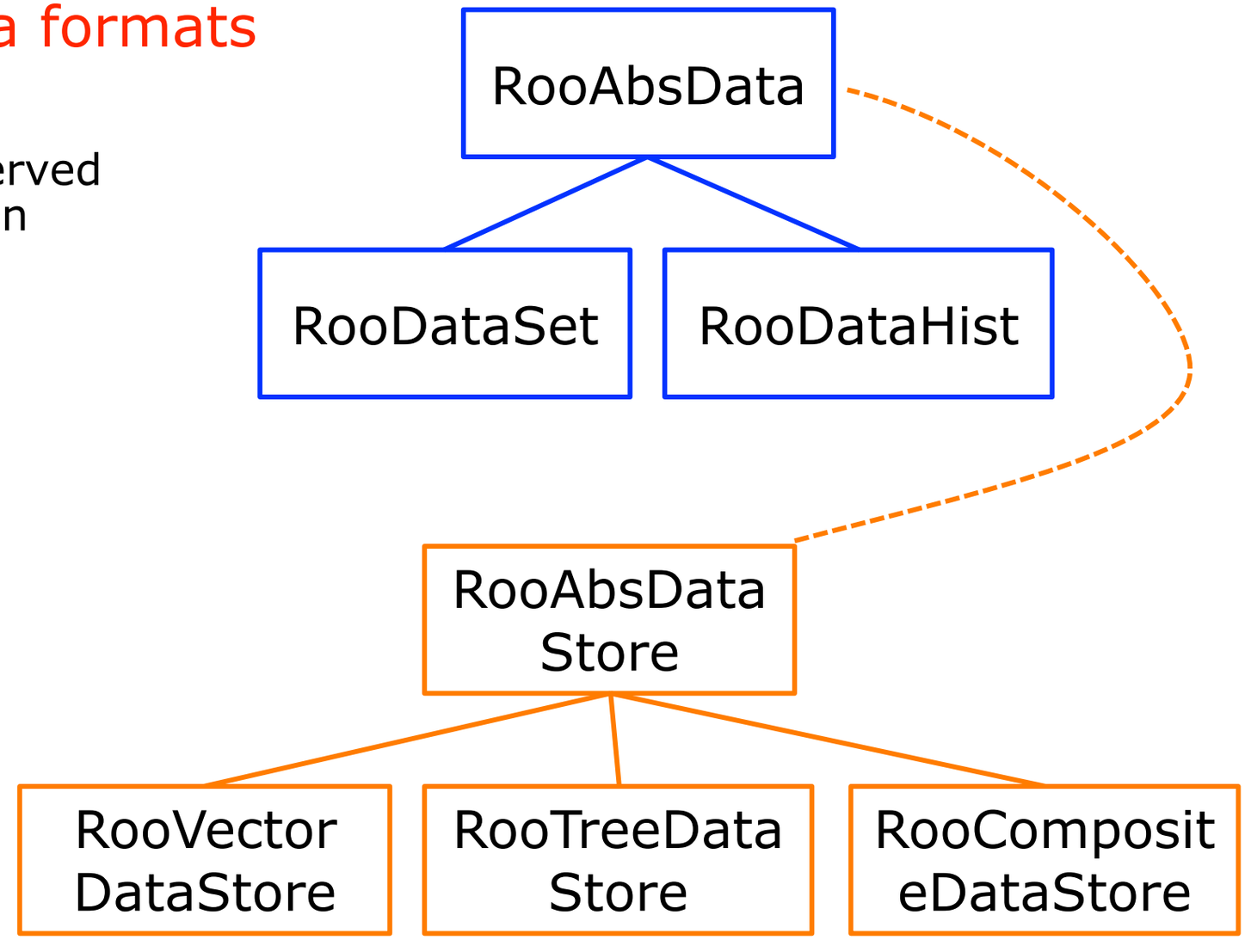
```
void func ( double* x, double* p, double* val )
2 {
    for ( i in range )
4       val[i] = exp( - p[0] * x[i] );
}
```

Listing 2: New vectorizable callback function for TF1

Optimization, parallelization, vectorization

- RooFit datasets have been migrated from TTree-based storage to std::vector based storage
  - Speedup in data reading inside likelihood by factor 40!
  - Essential TTree functionality of disk-based datasets not needed for likelihood fits
- Migrated class structure to decouple conceptual data formats from storage format
  - Backward compatibility preserved with custom schema evolution

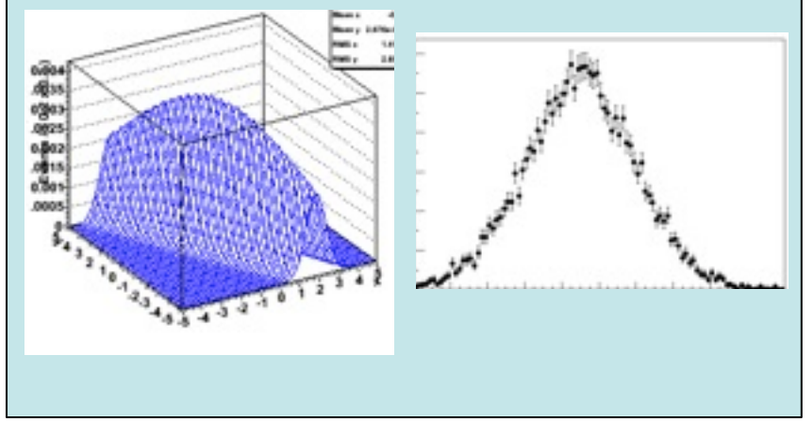




RooFit developments for LHC (Higgs analysis)

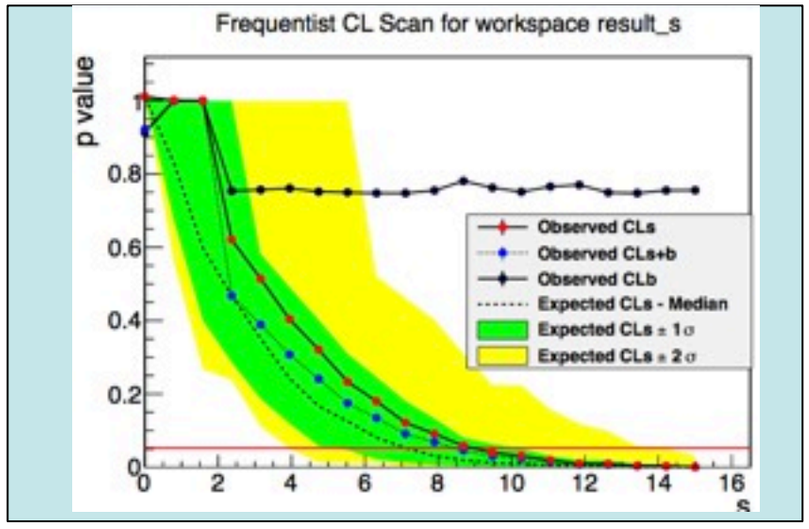
Class RooWorkspace

Simplify packaging and sharing of models



RooStats toolkit

Statistical tests based on likelihoods from RooFit models



HistFactory package

Constructing models from Monte Carlo templates