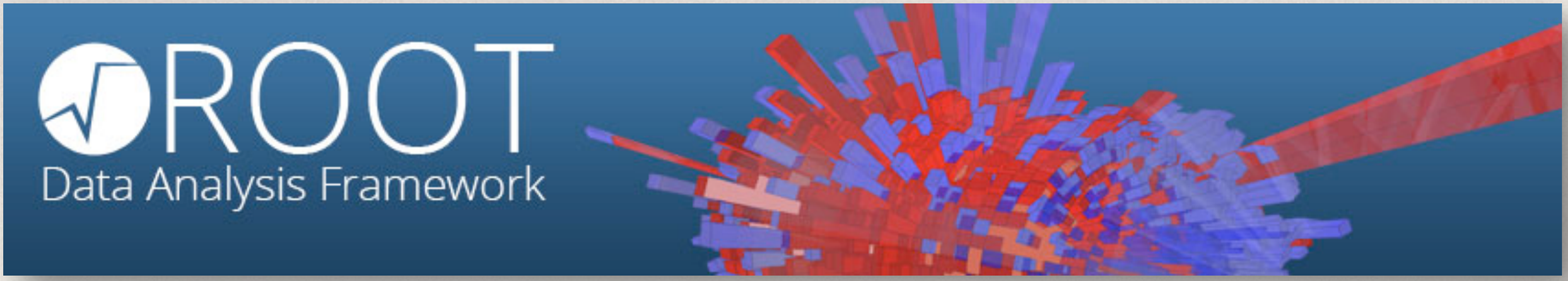


ROOT Development Roadmap

ACAT, 18-22 January 2016, UTFSM, Valparaíso (Chile)

Pere Mato for the ROOT team

Outline

- ROOT 6 current status
- Plans and ideas beyond version 6
  - Technical and Collaboration challenges
  - Main development axes
- Conclusions

What is ROOT 6?

- An evolution of ROOT5
  - "Like before, but better"
- Old functionalities preserved, new ones added
- ROOT6 and ROOT5 are compatible:
  - Old ROOT files are readable with the 6 version
  - New ROOT files are readable with the 5 version
  - Old macros can be executed with ROOT6 (if written in proper C++, see next slides)

CLING

- Replaces CINT: a radical change at the core of ROOT
- Based on LLVM and CLANG libraries.
  - Piggy back on a production quality compiler rather than using an old C parser
  - Future-safe - CLANG is an active C++ compiler
  - Full support for C++11/14 with carefully selected extensions
  - Script's syntax is much stricter (proper C++)
  - Use a C++ just in time compiler (JIT)
  - A C++11 package (e.g. needs at least gcc 4.8 to build)
- Will allow support for more architectures (ARM64, PowerPC64)

What is changing for the User?

implicit “auto”

```
root [0] h = new TH1F("myhisto","thetitle",10,0.,10.)
(class TH1F *) 0x7fa89c093bd0
root [1] h.Draw()
ROOT_prompt_1:1:2: error: member reference type 'TH1F *' is a
pointer; maybe you meant to use '->'?
h.Draw()
~^
 ->
```

Proper C++, good diagnostics

```
root [0] #include <map>
root [1] std::map<int,std::string> stringMap;
root [2] stringMap[0] = "zero";
root [3] stringMap[0]
(std::__1::map<int, std::__1::basic_string<char>,
std::__1::less<int>, std::__1::allocator<std::__1::pair<const int,
std::__1::basic_string<char> > > >::mapped_type &) "zero"[4]
```

C++

ROOT
Data Analysis Framework

What is changing for the User?

```
root [0] auto vars = gROOT->GetListOfGlobals()
(class TCollection *) 0x7f8df2b36690
root [1] for (auto && var : *vars) cout << var->GetName() << endl;
gROOT
gPad
gInterpreter
...
```

C++11

```
root [0] auto f=[](int a){return a*2;}
(class (lambda at ROOT_prompt_0:1:8) &) @0x109b7751c
root [1] f(2)
(int) 4

root [2] #include <random>
root [3] #include <functional>
root [4] std::mt19937_64 myEngine;
root [5] std::normal_distribution<float> myDistr(125.,12.);
root [6] auto myGaussian = std::bind(myDistr,myEngine);
root [7] myGaussian()
(typename __bind_return<_Fd, _Td, tuple<> >::type) 1.344781e
```

C++11

Python without Dictionaries

A.h

```
#include <iostream>
class A {
 public:
  A(const char* n) : m_name(n){}
  void printName() {std::cout<< m_name
                    << std::endl;}
 private:
  const std::string m_name;
};
int dummy() {return 42;}
```

python

```
>>> import ROOT
>>> ROOT.gInterpreter.ProcessLine('#include "A.h"')
0L
>>> a = ROOT.A('my name')
>>> a.printName()
my name
>>> ROOT.dummy()
42
```

- Great potential with many 3rd party libraries!!

ROOT
Data Analysis Framework

Not every thing has been great

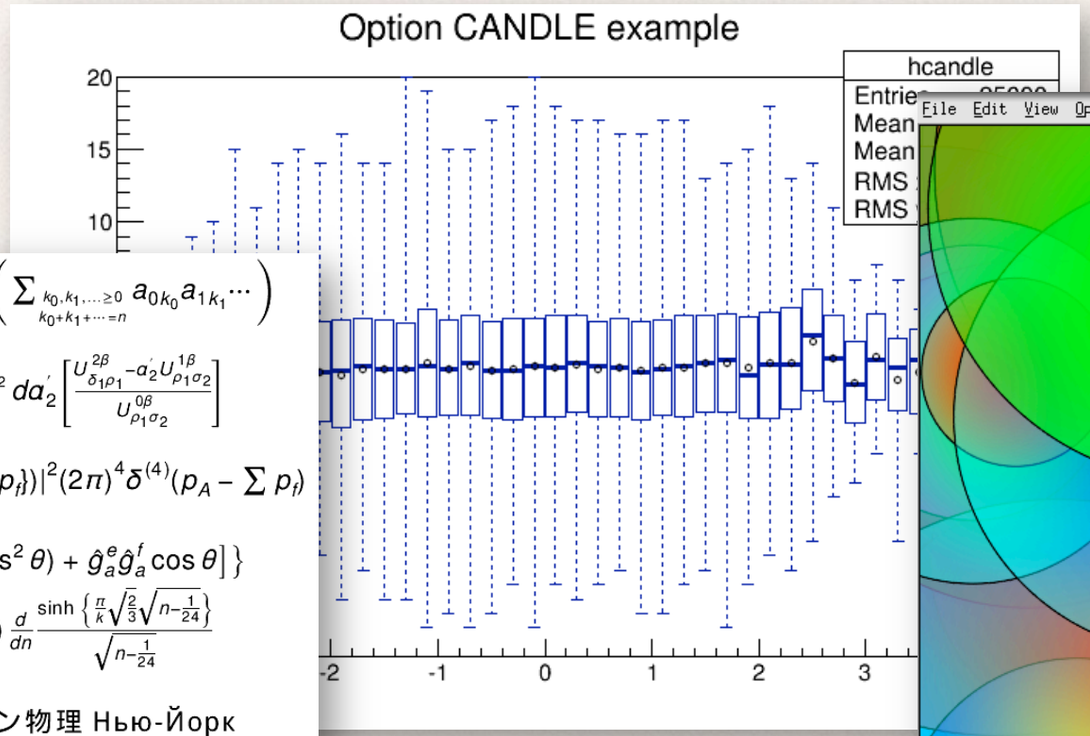
- Initial version of ROOT 6 (6.00) suffered from excess of memory utilization and slow startup time
  - Main cause: C++ PCMs no delivered on time by Clang, many headers files needed to be parsed at load time
  - Required a change in the design to minimize the number of files to be parsed: classes involved in I/O do not require their headers to be parsed
- Issue solved already in Autumn 2014
  - 6.02, 6.04, 6.06 series not affected!
  - The memory requirements are comparable (+- few %) to ROOT 5 for most of the use cases

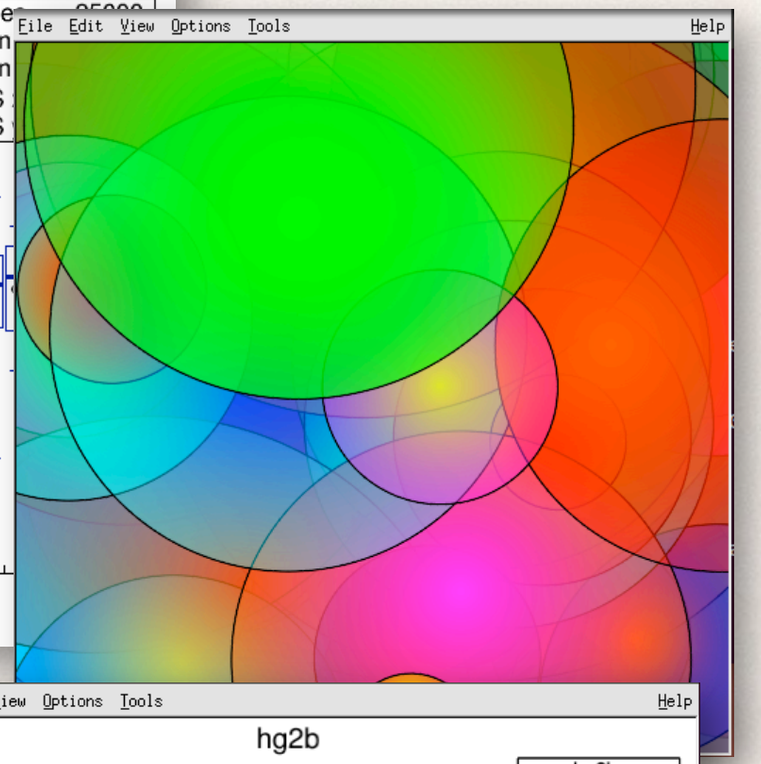
How did we get to this point?

- Integration with the LHC experiments' environments during the past 18 months
  - Weekly planning meetings to report progress and issues by the experiments
- And other non-LHC experiments (FairROOT, …)
- Integrating feedback from single users since May 2014
- Many fixes provided to the experiments for testing
- Usual patch and development branches available
- Overall very satisfactory feedback

ROOT 6 could not possibly have reached its current state without the help of all the LHC experiments

Graphics





$$\prod_{j\geq 0}\left(\sum_{k\geq 0} a_{jk}z^k\right) = \sum_{n\geq 0} z^n \left(\sum_{\substack{k_0,k_1,\ldots\geq 0 \\ k_0+k_1+\cdots=n}} a_{0k_0}a_{1k_1}\cdots\right)$$

$$W^{3\beta}_{\delta_1\rho_1\sigma_2} = U^{3\beta}_{\delta_1\rho_1\sigma_2} + \frac{1}{8\pi^2}\int_{a_1}^{a_2} da_2' \left[\frac{U^{2\beta}_{\delta_1\rho_1} - a_2' U^{1\beta}_{\rho_1\sigma_2}}{U^{0\beta}_{\rho_1\sigma_2}}\right]$$

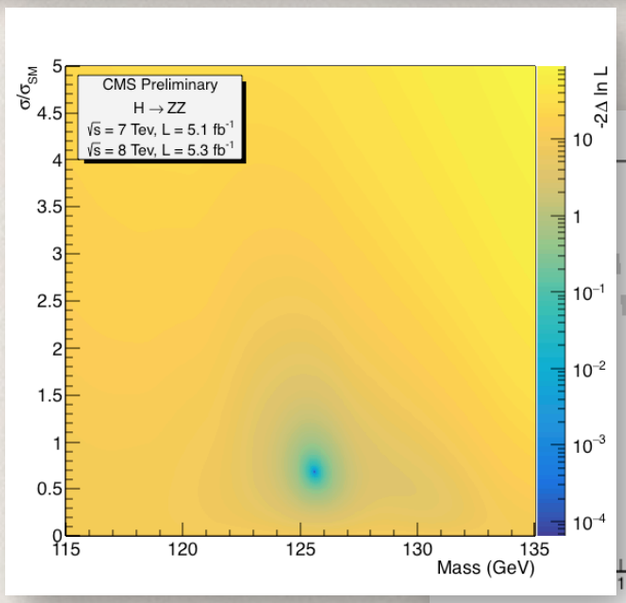
$$d\Gamma = \frac{1}{2m_A}\left(\prod_f \frac{d^3p_f}{(2\pi)^3}\frac{1}{2E_f}\right)|\mathscr{M}(m_A - \{p_f\})|^2(2\pi)^4\delta^{(4)}(p_A - \sum p_f)$$

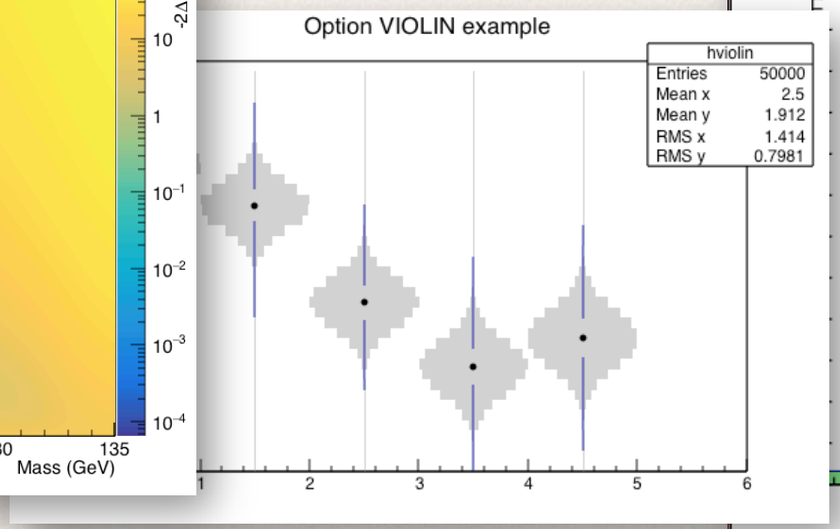
$$4\mathrm{Re}\left\{\frac{2}{1-\Delta\alpha}\chi(s)\left[\hat{g}^e_v\hat{g}^f_v(1+\cos^2\theta) + \hat{g}^e_a\hat{g}^f_a\cos\theta\right]\right\}$$

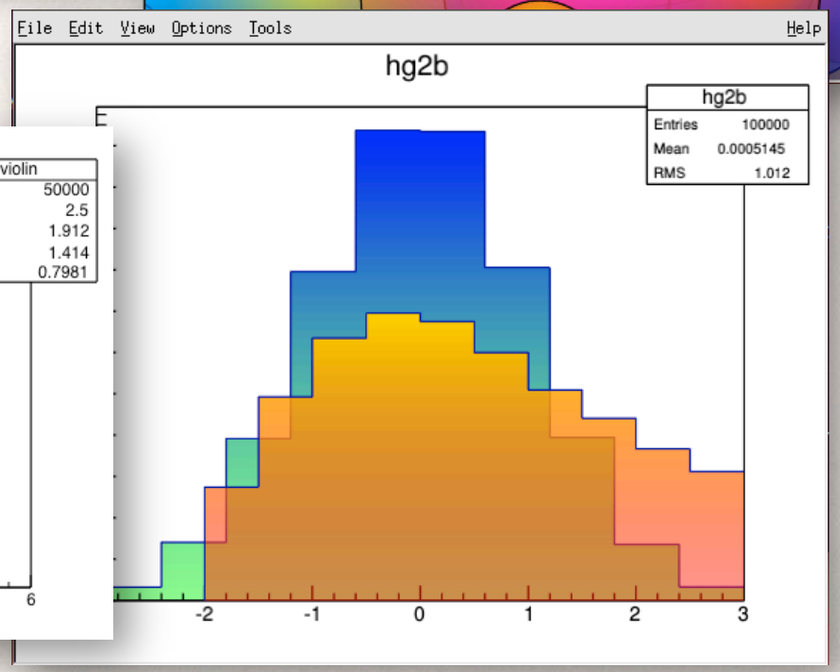
$$p(n) = \frac{1}{\pi\sqrt{2}}\sum_{k=1}^{\infty}\sqrt{k}A_k(n)\frac{d}{dn}\frac{\sinh\left\{\frac{\pi}{k}\sqrt{\frac{2}{3}}\sqrt{n-\frac{1}{24}}\right\}}{\sqrt{n-\frac{1}{24}}}$$

$$\frac{(\ell+1)C_\ell^{TE}}{2\pi} \quad \mathbb{N}\subset\mathbb{R}$$

RHIC スピン物理 Нью-Йорк







ROOT
Data Analysis Framework

Graphics: bye rainbow palette

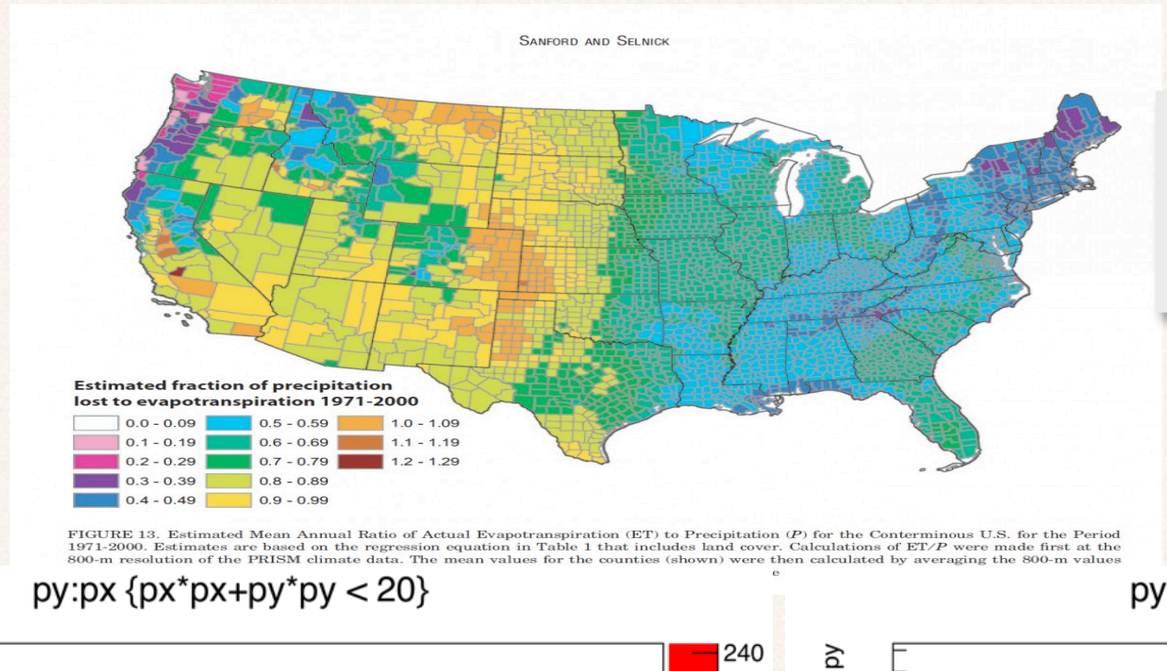
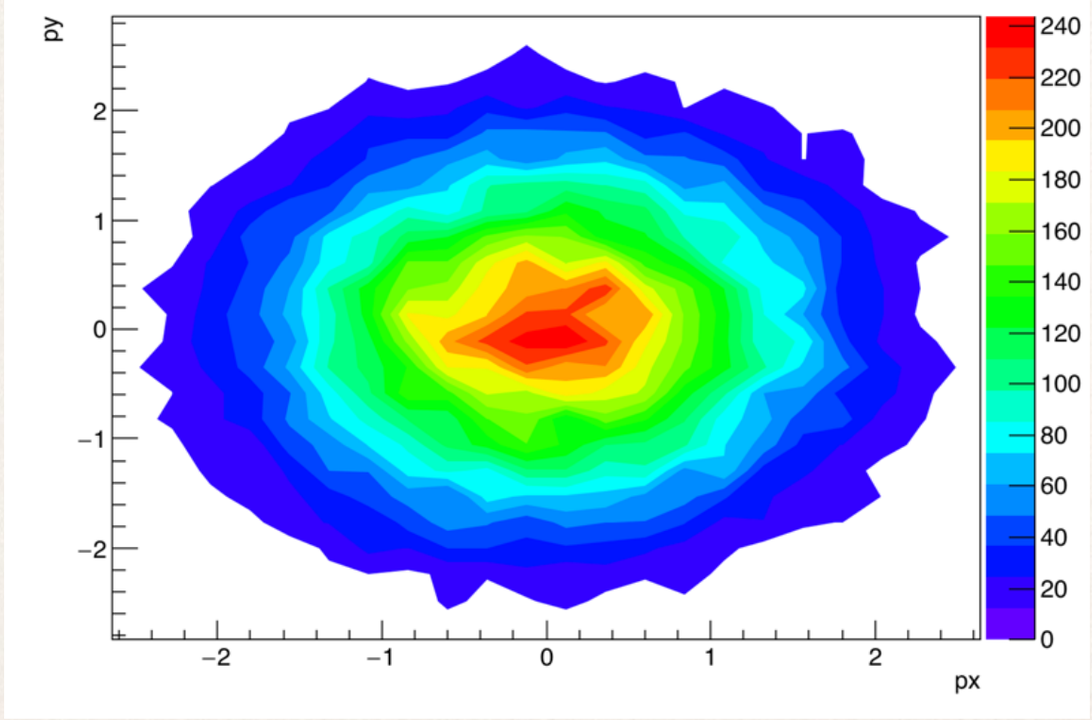
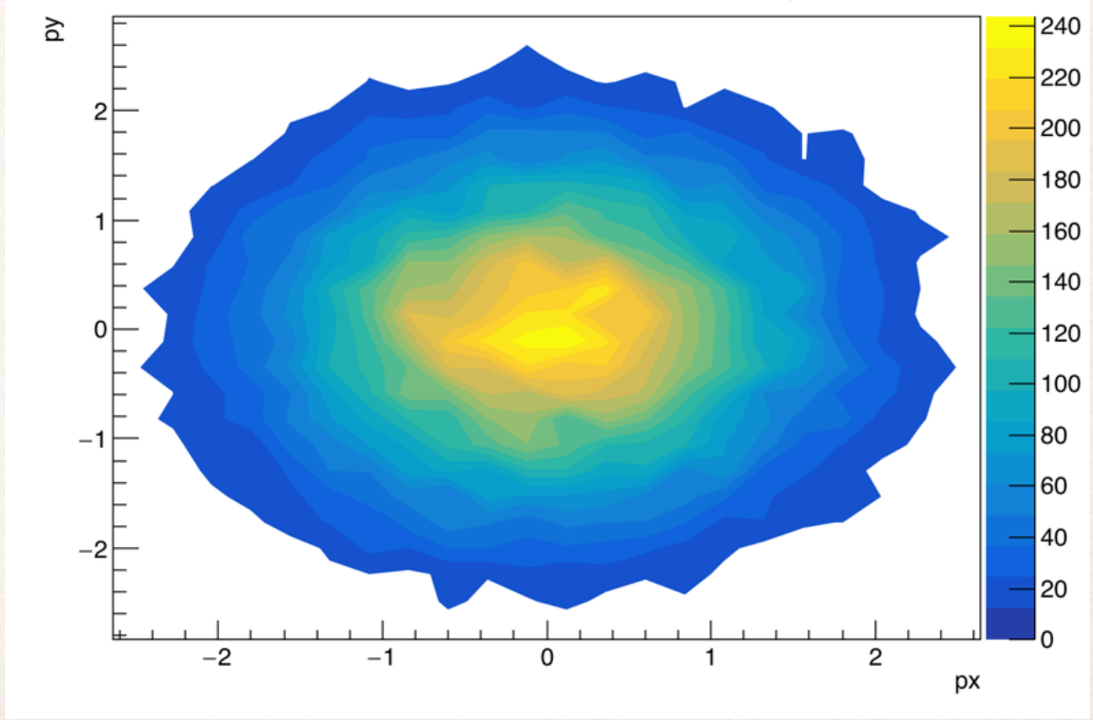


FIGURE 13. Estimated Mean Annual Ratio of Actual Evapotranspiration (ET) to Precipitation (P) for the Conterminous U.S. for the Period 1971-2000. Estimates are based on the regression equation in Table 1 that includes land cover. Calculations of ET/P were made first at the 800-m resolution of the PRISM climate data. The mean values for the counties (shown) were then calculated by averaging the 800-m values

the new palette is the default!





ROOT
Data Analysis Framework

New TFormula

- Example of using the new JIT capability of LLVM/CLANG
- Pre-parsing of expressions (as before) but now using a real compiler
- I/O backward compatibility has been preserver

```
auto f = new TFormula(“F”,”[0]+[1]*x”);

auto f = new TFormula("F","[](double *x, double *p)
                          {return p[0]+p[1]*x[0];}”,1,2);
```

ROOT-R Interfaces

- R functions available in ROOT
- Wrapper classes for R data objects (e.g. TRDataFrame)
- R plugins for minimization
- R plugins for TMVA
- decision tree libraries (xgboost), neural network, and support vector machine packages

```
template<class T> T fun(T x) { return x+x; }

auto r = TRInterface::Instance();
r[“func”] << fun<TVector>;
r << “print(fun(c(0.5,1,1.5,2,2.5)))”;

1 2 3 4 5
```

ROOT
Data Analysis Framework

MultiProc package

- Developed a new lightweight framework for multi-process applications
  - Inspired by the Python *multiprocessing* module
  - Idea to **re-implement Proof-Lite** using it
  - Distribute work to a number of fork()'d *workers,* then collect results
    - Main advantage: workers have access to complete 'master' state

```
TPool pool(8)

auto result = pool.Map(fun, args);

auto result = pool.MapReduce(fun, args, redfun);
```

defaults to # of cores

C/C++ function loaded macro std::function lambda

std::container initializer list TCollection& unsigned N

std::vector or TObjArray

Reduce function

ROOT Data Analysis Framework

Build and Testing System

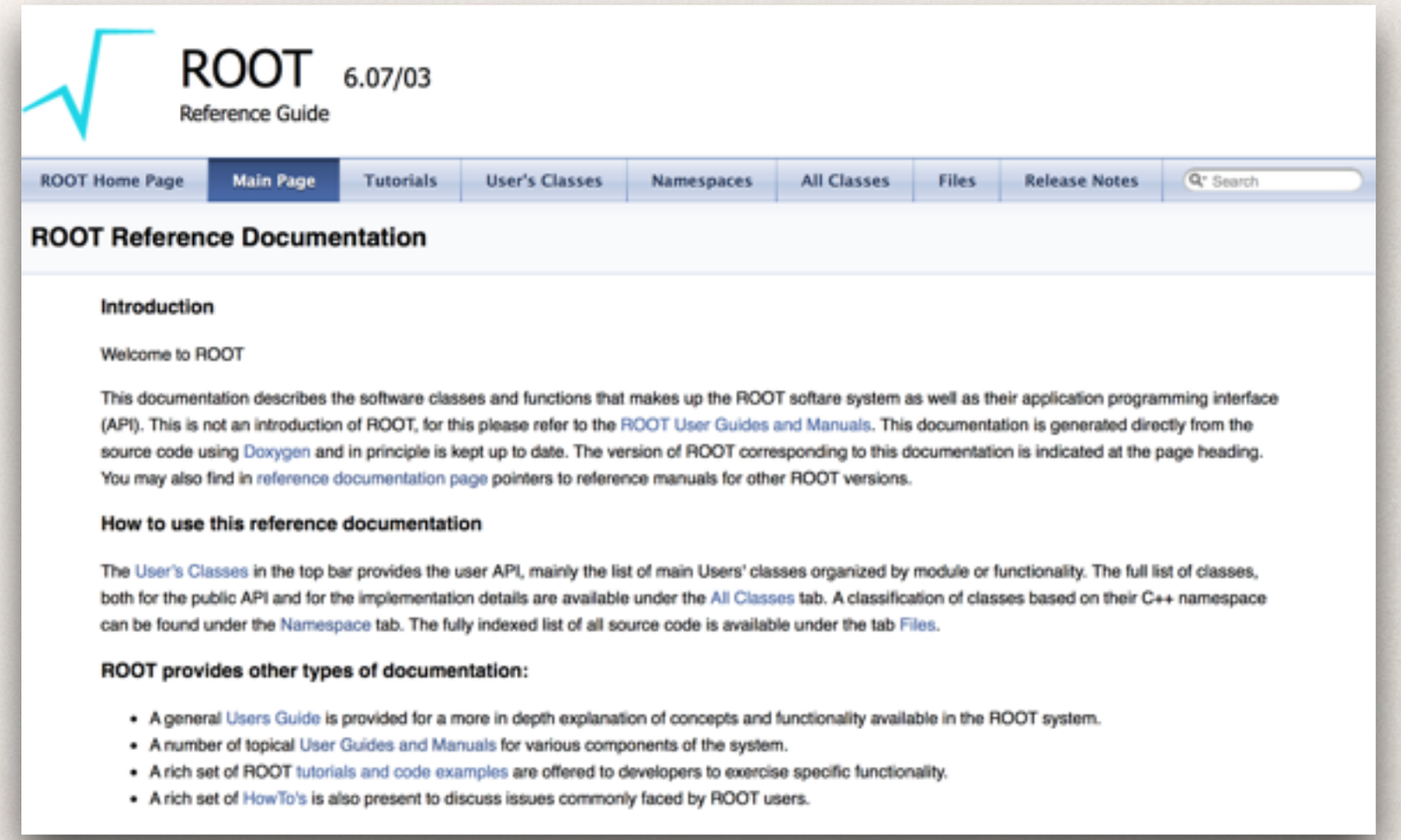
- ROOT uses the **CMake** cross-platform build-generator tool as a primary build system
  - Native windows builds, support for many build tools: GNU make, Ninja, Visual Studio, Xcode, etc
  - See instructions at https://root.cern.ch/building-root
    - Classic **configure/make** will still be maintained, but it will not be upgraded with new functionality, platforms or modules.
- Unit and Integration tests (~1400) have been migrate to **CTest**
- Binary installations are packaged with **CPack**
- Nightly and Continuous integration builds are automated and scheduled with **Jenkins,** as well as all the release procedures

ROOT 6 Completion

- **Introduction of PCMs** (Pre-compiled Modules)
  - Minimize parsing of headers (the biggest source of extra memory consumption)
  - Avoid to need of headers deployment
  - To achieve a smooth integration of PCMs to the experiments software systems will require some work
    - Still some technical decisions to be taken
- **Windows support**
  - New versions of LLVM expected to work on Windows
- Aiming for completion for version 6.08 in May 2016

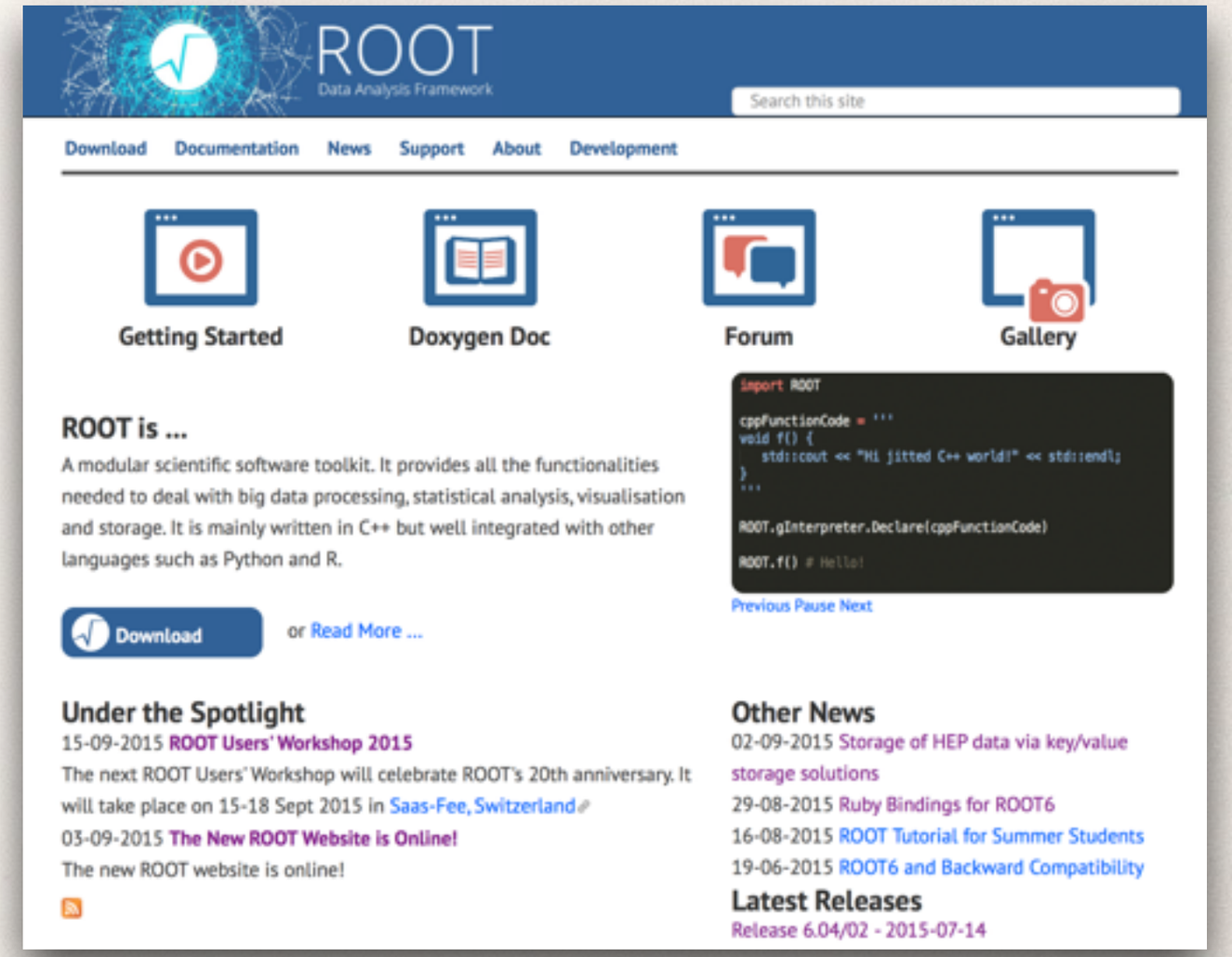
Doxygen Reference Guide

- ROOT Reference Documentation is now generated with **Doxygen**
  - **https://root.cern.ch/doc/master**
- Translated and updated hundreds of thousands of documentation lines to doxygen
- Up-to-date pictures still generated from ROOT macros
- Integrated tutorials



New ROOT Web

- ROOT website migrated to Drupal 7
  - hosted in CERN web infrastructure
- Took the opportunity to revise the content, to revise the organization and to give a new look



Development Beyond ROOT 6

Technical Challenges

- ROOT is 20 years old, and some parts **require re-engineering and modernization**
- Exploit modern hardware (many-core, GPU, etc.) to boost performance
- Modernize implementations (C++11 constructs, use existing libraries, etc.)
- Need to reflect on the needs and eventually solve the backward / forward compatibility

Collaboration Challenges

- ROOT is 20 years old, and requires the collaboration of the community to **ensure evolution and sustainability**
- We would like to facilitate contributions to ROOT without engaging our responsibility in the maintenance and user support
  - layered software modules or plugins that can bring new functionality to the end-users
  - e.g. systems like Jenkins/Drupal/R provide a platform for developers to contribute in an easy manner
- The nature of the contributions can be on extending existing functionality and later on the changes in the core functionality

Development Main Directions

- **Cling Interpreter and its full exploitation**
  - C++11/14, JIT compilation opens many possibilities (automatic differentiation, improved interactivity, etc.)
- **Modern C++ interfaces**
  - Explore better C++ interfaces making use of new standards (C++14, C++17)
- **Parallelization**
  - Seek for any opportunity in ROOT to do things in parallel to better exploit the new hardware (e.g. Ntuple processing, I/O, Fitting, etc.)

Development Main Directions (2)

- **Packaging and modularization**
  - Incorporate easily third party packages (e.g. VecGeom in TGeom)
  - Build/install modules and plugins on demand. Facilitate contributors to provide new functionality
- **Re-thinking user interface**
  - Explore new ways to provide thin-client web-based user interfaces
  - Javascript and ROOTbooks
- **ROOT as-a-service**
  - Thin client plugged directly into a ROOT supercomputing cloud, computing answers quickly, efficiently, and without scalding your lap

Cling Interpreter

- Upgrade to latest LLVM/CLANG
  - Waiting until it supports new ABI (GCC 5)
- Optimizations and improvements
  - reduce and CPU and memory consumption
  - transparently be able to create functions and the ROOT prompt
  - tab-completion based on multi-interpreters
- PCM (Pre-Compiled Modules)
  - Efficient way to store reflection information with the potential to implement functionality equivalent o ROOT dictionaries
  - Replace the today unique PCH by a number of PCMs

New C++ Interfaces (ROOT 7)

- We want to make ROOT simpler and more robust
- Many interfaces can be improved with C++14, 17
  - Ownership, type safe containers, string options
  - **Improved user productivity**, by dramatically reducing memory errors, wrong results, etc.
- The price to pay is to **break backward C++ interface compatibility** as announced at ACAT 2014
  - During the transition period new classes co-exists with old classes (namespaces)
  - Old data will always be readable
- Important to get feedback from users and developers
  - Bi-weekly meetings with users from experiments
  - started discussing iteration and ownership issues
- Starting with histograms + visualization, and TFile

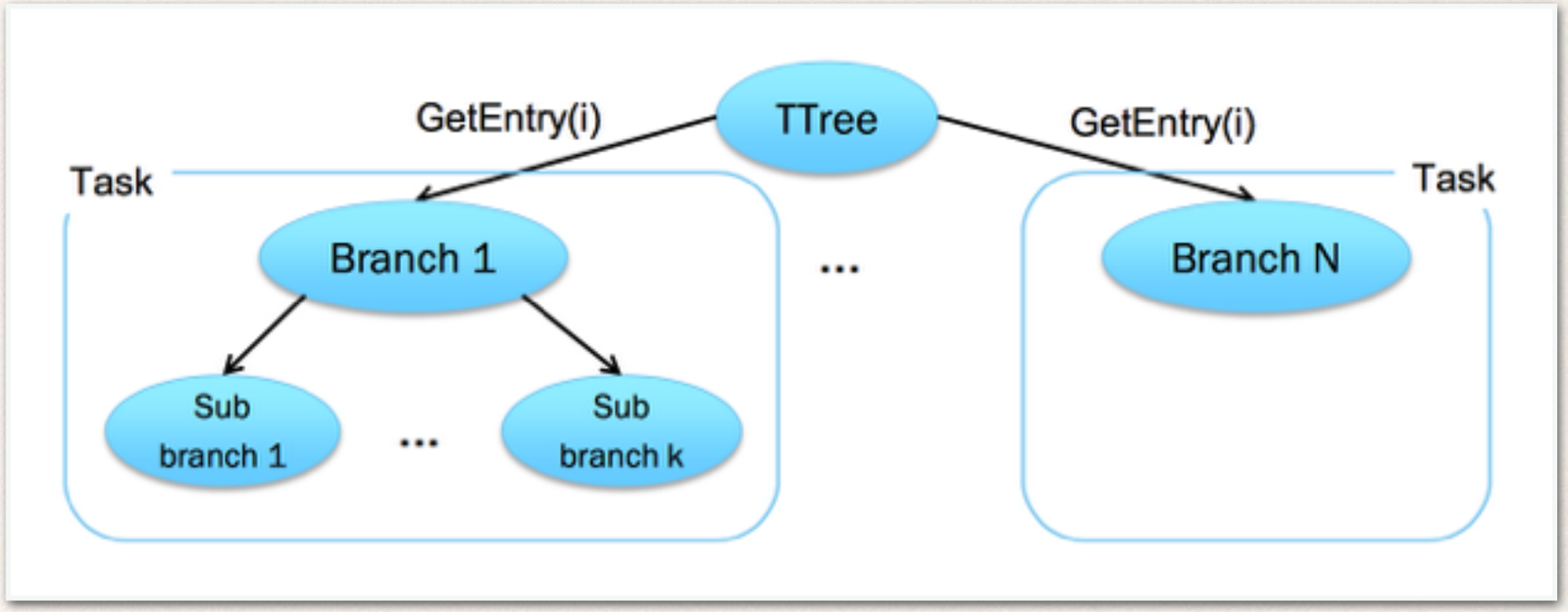
Parallelization

- Re-engineer Proof-Lite for executing parallel tasks in both **multi-process** and **multi-thread**
  - Same interface between the multi-threading and multi-processor solutions
- Prototype solution(s) for a number of use cases:
  - Histogram/ntuple filling, TTree processing (TTreeDraw), I/O pipeline, Minimization/Fitting, etc.
- Make parallelization transparent when possible, provide user-friendly means otherwise
- Solve problems for **merging efficiently the output objects** produced by the parallel tasks: (histograms, trees, etc….)
- Introduce **thread-safety** where needed (e.g. I/O)

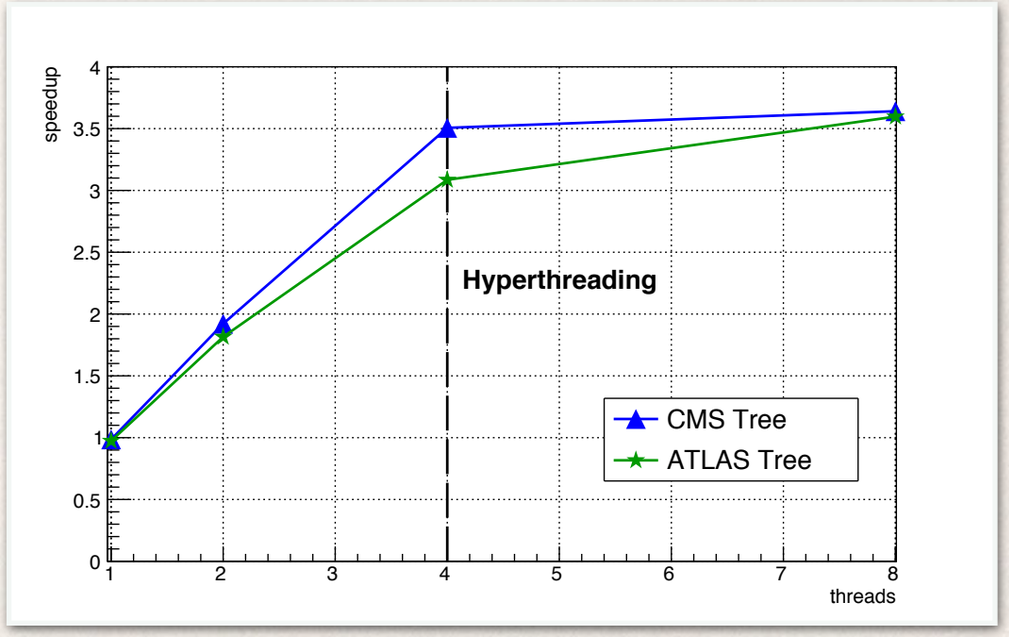
Parallelization: Multi-process

- Improve the new released **core/multiproc** package
  - Complete support for TSelector and provide integration with TTree::Draw
  - Provide detailed documentation and more complex examples
  - Extension of the **multiproc** interfaces to a cluster of machines (à la iPython parallel)
- PROOF in maintenance mode
  - No major development foreseen

Parallelization: Multi-threading



- Implemented parallel TTree reading using a "task programming model" (e.g. TBB)
  - speeding up the TTree:GetEntry(i)
- Continuing with other use cases:
  - Three::Draw, TTree::Process(lambda)
  - Histogram fitting, RooFit, etc.



ROOT
Data Analysis Framework

Modularization and Packaging

- ROOT depends of a number of external libraries
  - strongly: X11, freetype, zlib, pcre, …
  - weakly: if the optional component is enabled
- Uniform treatment of external dependent libraries
  - Bundle/unbundle modes controlled by build options
- Converge on a **factorization of the logical components of ROOT** and produce an analysis of the current interdependencies among them
- Develop model for **building/installing modules on demand** and evolve ROOT into BOOT (à la R)
  - Essential to facilitate contributions

Python Ecosystem

- Python3 fully supported
  - Few pending issues to be fixed
- Release PyROOT as a genuine Python package
  - Co-existing within Python2 and Python3 installations
  - build/install based on Python package manager
- Python plugins for TMVA (using scikit-learn package)
- Prototype tree/ntuple analysis which adopts a *Data Frame*-like interface
  - Interoperation with popular scientific Python packages
  - Learn from interfaces used by crowds of data scientists

I/O Evolution

- Performance
  - Better support concurrent I/O operations
  - Code implementation optimization
  - Optimization via change in file format (endianness, memory layout, etc.)
- New Features
  - Support for C++11 classes (std::array, pointers, etc,)
  - Open up the interface of TFile to be able to make use of key-value storage (kinetics)
  - Support for JIT-ted collection proxies to enable transparently the serialization of collections

Geometry

- Evolve the **TGeo package** and its derived navigation interfaces (VMC) to support for vectorized navigation based on the **VecGeom package**
  - Add VecGeom library as optional external module
  - Phase-1: Implementation of a TGeoShape-derived bridge class (TGeoVGShape) delegating the navigation interface to the VecGeom solid
  - Phase-2: Implementation of a VecGeom-aware navigation interface that can redirect the current navigation API of TGeoManager / TGeoNavigator to native VecGeom navigators

Math Libraries

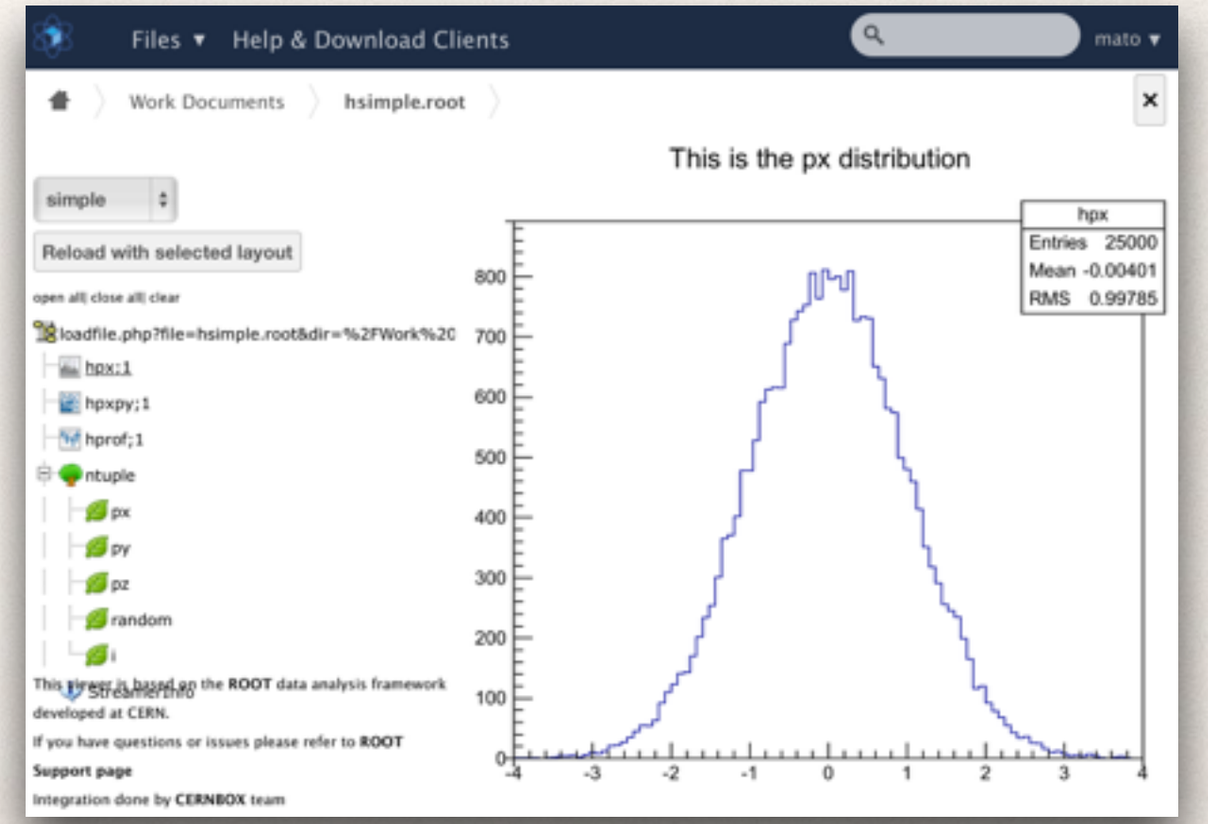
- Exploit **vectorization** in evaluations of functions (e.g. for fitting)
- Use multi-thread and multi-process **parallelization** in fitting and statistical studies (e.g. RooStats)
- Fully integrate new pseudo-random number generator (**MIXMAX**) better suited for parallel environment and based on strong theoretical grounds (Kolmogorov K-system theory)

Machine Learning

- Active participation to the Inter-Experimental LHC Machine Learning Working Group IML launched recently
- Modernize and improve TMVA
  - New and more flexible design
  - Integration of new methods (cross validation, new deep learning neural network)
  - Optimization and parallelization of algorithms
  - Facilitate interoperability with other machine learning software packages

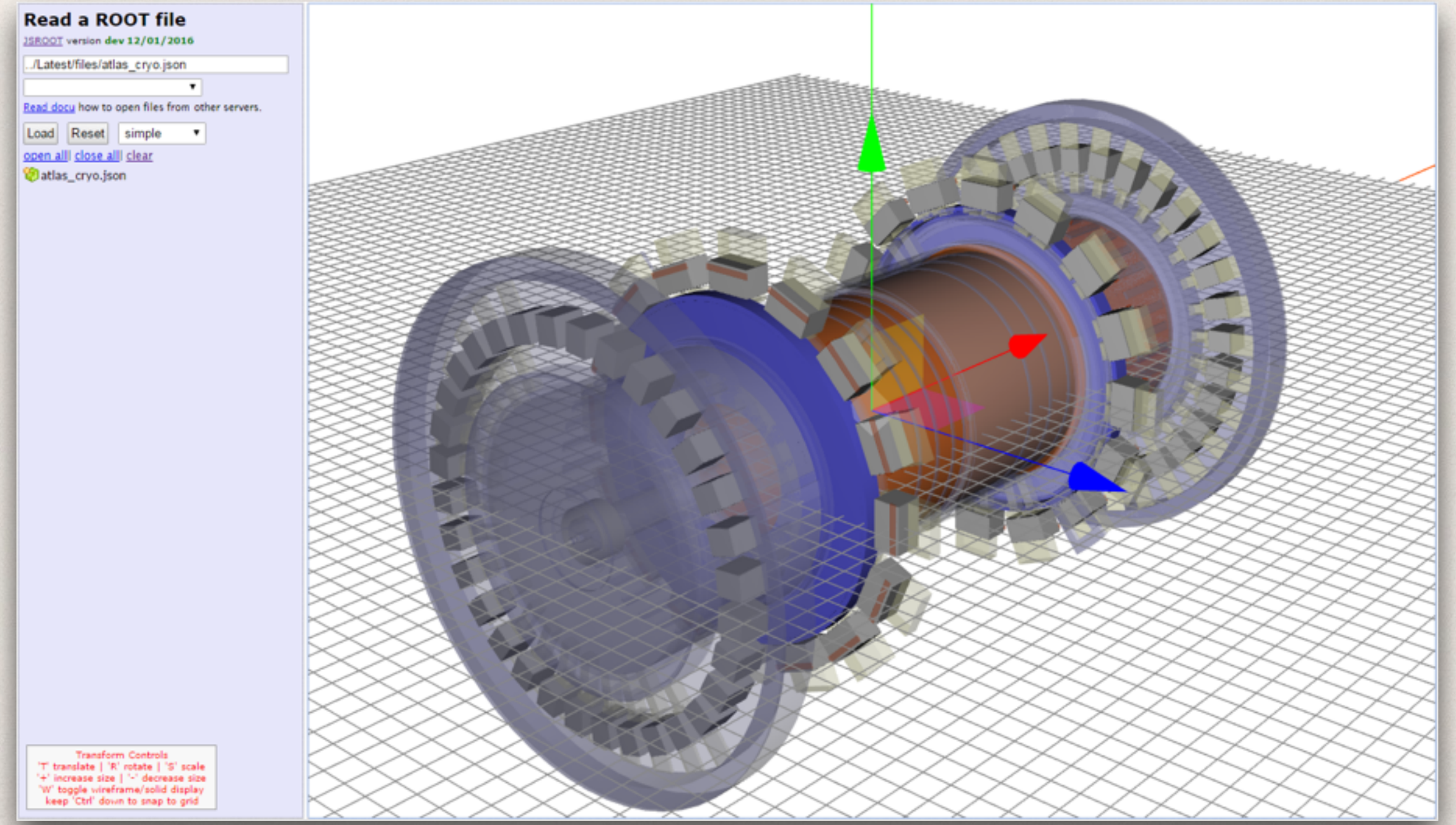
Rethinking UI

- Exploring new ways to provide thin-client web-based user interfaces
- Increase interactivity using modern web technology (javascript) in a client-server model
  - No need to install anything in the client side
  - 3D geometry viewer
- Building on the HttpServer and JavaScript interface (JSROOT)
- CERNBox Example

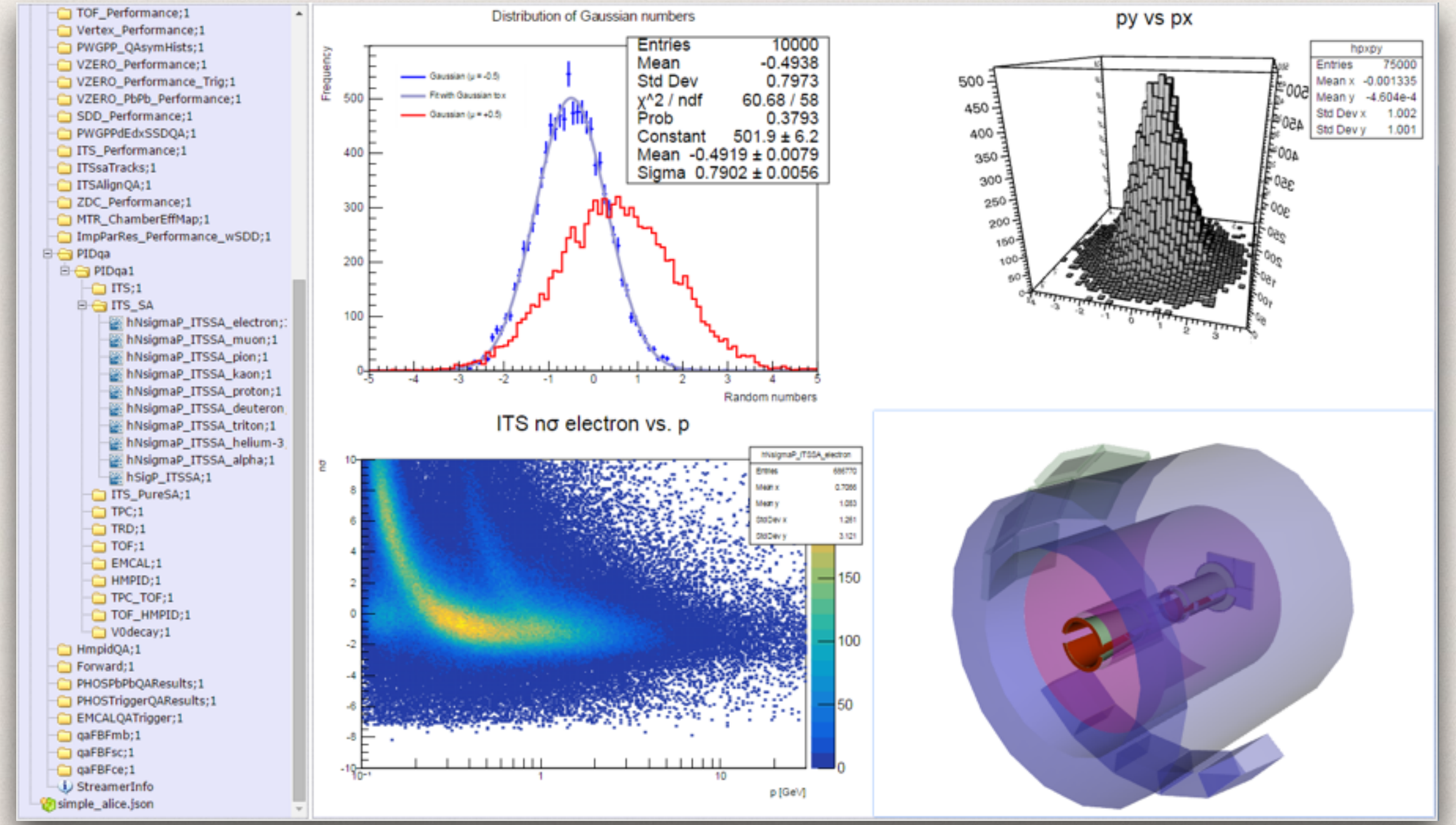


ROOT
Data Analysis Framework

JavaScript: 3D Viewer

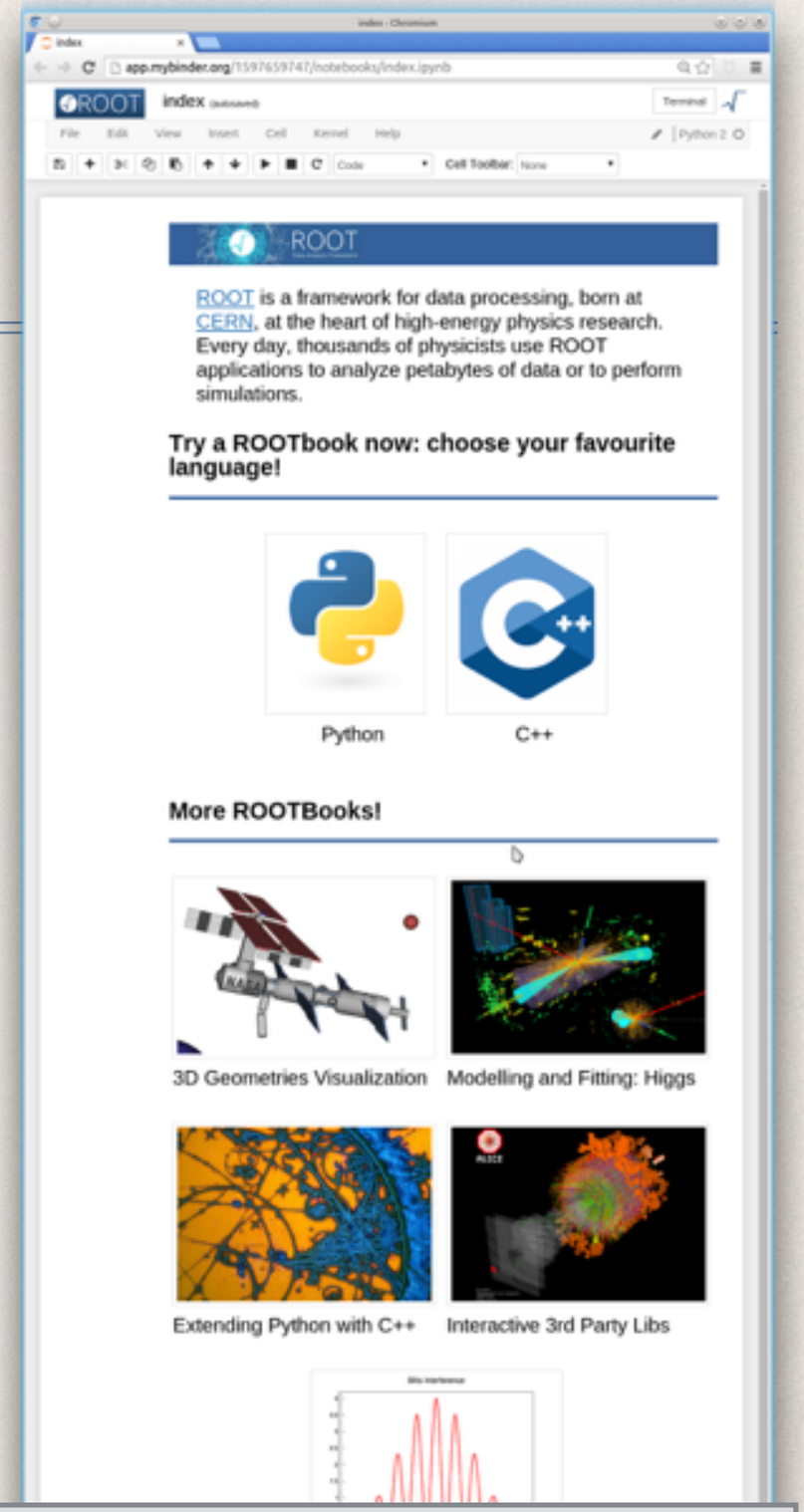


JavaScript: New plots



ROOTbooks

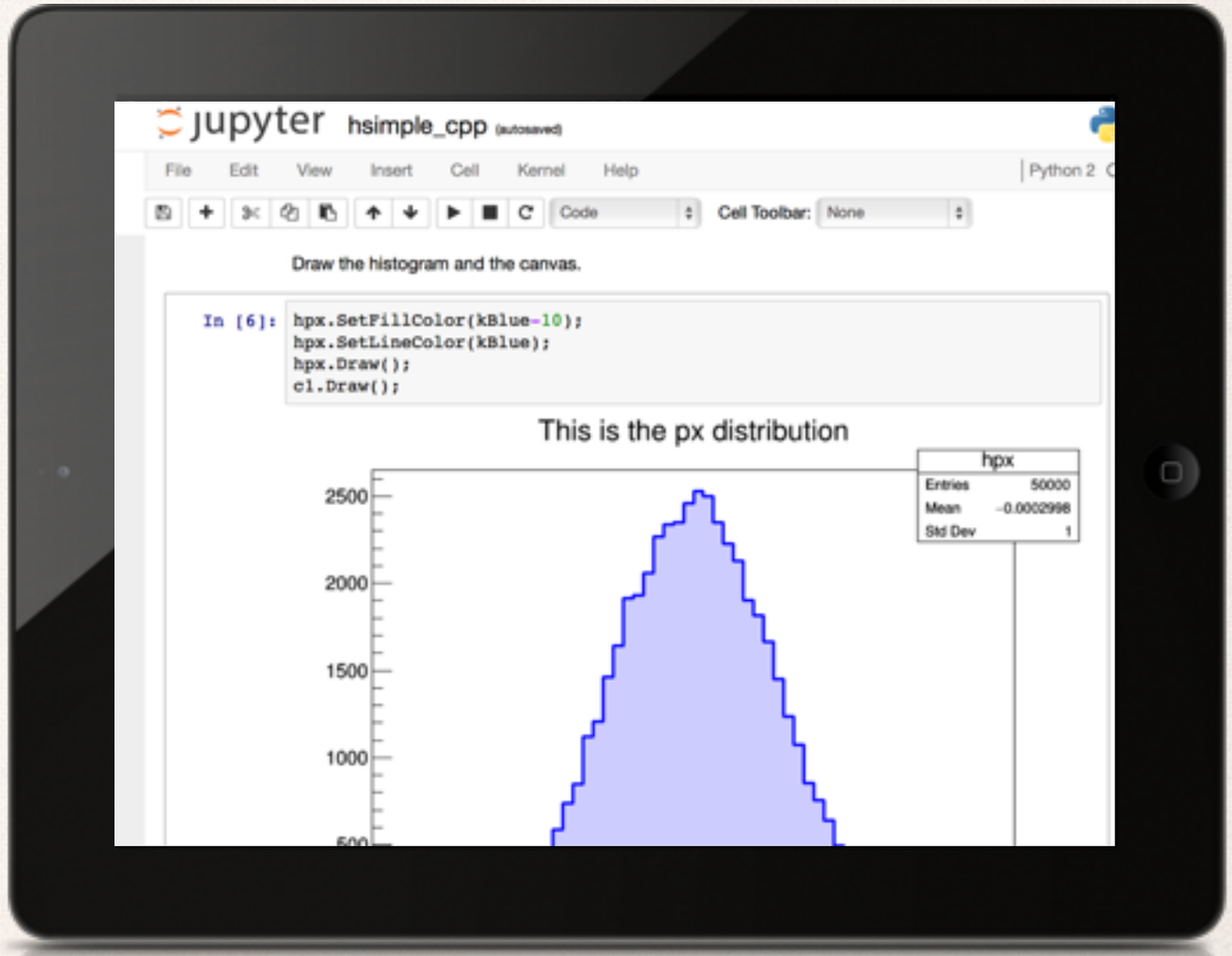
- The integration of ROOT with the **Jupyter notebook technology** (ROOTBook) is well advanced
  - Ideal for training material
  - Possible way to document and share analysis
- Still some work to do in the following areas:
  - Disseminate the use of ROOTbooks
  - Make the JS visualization the default
  - Add R



Take a tour with Binder (no account needed)
http://mybinder.org/repo/cernphsft/rootbinder

ROOT as-a-Service

- Combines naturally the work on **parallelization** to exploit many cores and nodes together with the new **web-based interface** to provide a modern and satisfying user experience
  - Build on top of cloud security, storage and compute services
- Working towards a **Pilot Service** that is able to serve requests for a medium number of users (O(100))
  - Implemented using the CERN palette of IT services



ROOT
Data Analysis Framework

ROOT Versions

- **5.34 - current production**
  - ROOT 5 is frozen except for critical bug fixes
- 6.02
  - First functionally complete ROOT 6 version. Superseded by 6.04
- 6.04
  - Used by the LHC experiments in production
- **6.06 - current production**
  - Released beginning of December 2015
  - 6.07/02 - development release
- 6.08 - hoping for PCMs and Windows support
  - Scheduled for May 2016

Collaborations

- ROOT has currently many external contributors:
  - in 2015 we had commits from 57 different authors
- Active Collaborate with us and Ideas web page
- Interest of other projects to collaborate with ROOT
  - DIANA
    - Data Intensive ANAlysis, 4-year NSF funded
    - Focus on analysis software, including ROOT and its ecosystem
    - Three primary goals: performance, interoperability, support for collaborative analysis
  - Other initiatives in the pipeline
- Need to integrate these collaborations and achieve a coherent program of work

Conclusion

- ROOT6 is used in production by most LHC experiment and others
- Long list of development ideas following the following axes:
  - Modernization of C++ API
  - Parallelization with threads and processes
  - Packaging and modularization
  - Exploiting Python ecosystem
  - Machine Leaning
  - ROOTbooks
  - ROOT as-a-service
- Collaborations and contributions most welcome