ROOT
Data Storage
&
Analysis Framework

Stanford, XLDB2015
René Brun / CERN
ROOT Users' Workshop 2015

workshop

Your ROOT turns 20!

We would like to invite you to the ROOT Users' workshop in Saas-Fee, Switzerland, from September 15-18 (Tuesday to Friday), 2015 - to celebrate its birthday, discuss its present and cook up its future!

We will have several extraordinary presentations, for instance by Bjarne Stroustrup on how to write modern, good C++. The majority will include live discussions with users and all the ROOT developers, in an inviting setting.

Read more
HEP (High Energy Physics)  
Observable Details

High energies may be produced by accelerators or cosmic rays.

atom

proton

quark
HEP & related fields

• About 15000 HEP physicists in the world, a large fraction being experimentalists.

• Large laboratories
  o USA: FNAL, BNL, SLAC, JLAB..
  o Japan: KEK
  o Europe: CERN, DESY

• Astrophysics: growing connections with HEP
  o In space: Fermi, AMS, Plank, ...
  o On ground: LSST, HESS, IceCube, Auger, SKA, ...

• Neutrino Physics: Kamiokande, FNAL, Gran Sasso, Dahia Bay..

• Gravitational Waves: LIGO, VIRGO

• Medical Physics

• Hadron Therapy

R.Brun: ROOT: data storage & analysis framework

May 22 2015
The Large Hadron Collider (LHC) is being built in a circular tunnel 27 km in circumference. The tunnel is buried around 50 to 175 m. underground. It straddles the Swiss and French borders on the outskirts of Geneva.
LHC
A simulated event or real event?
HEP Software: main fields

- Math libs: linear algebra, statistics, ...
- Histogram packages, minimization, visualization, UI
- Data models, data storage, data access
- Data analysis tools
- Detector simulation
- Event generators
- DAQ systems
- Grids/Clouds technology
- Software engineering, quality, packaging/distribution, documentation

May 22 2015
R. Brun: ROOT: data storage & analysis framework
Systems today

- **End user Analysis software**: 0.1 MLOC
- **Experiment Software**: 5 MLOC, 1000+ nnn
- **Frameworks like ROOT, Geant4**: 6 MLOC, 20 + 500
- **OS & compilers**: 20 MLOC
- **Clusters of multi-core machines**: 1000x8

- **Networks**: 10 Gbit/s
- **Disks**: 10 PB
- **Developers**: nnn

**Hardware**
- RAM: 16 GB
- GRIDS
- CLOUDS
1994: turning point

- **SSC** project in Dallas is stopped
- **LHC** project approved
- Many refugees from the SSC have a strong voice in various software workshops, in particular Object Oriented Data bases become fashionable because **OO** style programming is also becoming fashionable.
- RDRC projects launched at CERN
  - RD44: **GEANT4**
  - RD45: **Objectivity** DBMS
- Linux is coming to HEP (only a few evangelists)
- Pentium Pro revolution ➔ end of MPP era
Languages in HEP

- Fortran
- C/C++
- Python
- Java

R.Brun: ROOT: data storage & analysis framework
Problem: How to develop/maintain large software systems to be used over many decades? Centralism (eg at CERN) vs open/international teams
ROOT: a Framework and a Library

- User classes
  - User can define new classes interactively
  - Either using calling API or sub-classing API
  - These classes can inherit from ROOT classes
- Dynamic linking
  - Interpreted code can call compiled code
  - Compiled code can call interpreted code
  - Macros can be dynamically compiled & linked

This is the normal operation mode

Interesting feature for GUIs & event displays

Script Compiler
root > .x file.C++
The Libraries

- Over 2000 classes
- 3000,000 lines of code
- CORE (12 Mbytes)
- Green libraries linked on demand via plug-in manager (only a subset shown)
- About 120 shared libs
Interpreters

- From 1995 to 2014 (version 5.34) the command line and script interpreter was based on **CINT** (written by Masa Goto from Agilent Technologies Japan). **CINT** included only a subset of **C++91**
- Since 2014 (version 6) ROOT interpreter is based on **CLING** (derived from **CLANG/LLVM** from Apple).
  - **CLING** provides a FULL **C++** interface.
  - A Python interface (**PYROOT**) is also provided.
Dictionaries

All the necessary information for the interpreter and I/O is here
Dictionaries (2)

• In the past, users/developers could implement any type of data structure (with language limitations), but the interpreters (command line) or I/O systems like ZEBRA, HDF, HDF5, etc were allowing only very specific formats.

• When developers are hundred, thousands, this limitation becomes impossible.

• Interpreters and I/O must be able to support any data model from large projects, having in mind a continuous evolution of this data model(s) across several decades.
Interpreter & Compiler integration

- **execute file** `script.C`
- **execute function** `DoSomething`
- **compile file** `script.C` and execute it
- **compile file** `script.C` if file has been modified. execute it
- **same from compiled or interpreted code**
Graphics, UI
2-D Graphics

- New functions added at each new release.
- Always requests for new styles and new ways to display data.
- `ps, pdf, svg, gif, tex, jpg, png, c, root`, etc

R.Brun: ROOT: data storage & analysis framework
Graphics (2D-3D)

- "LEGO"
- "SURF"
- TGLParametric
- TH3

R.Brun: ROOT: data storage & analysis framework
see $ROOTSYS/tutorials/geom

- R.Brun: ROOT: data storage & analysis framework
R. Brun: ROOT: data storage & analysis framework

May 22 2015
The I/O, Data Storage and queries
Data volume

But the data volume is only one side of the problem
The way the information is organized and indexed is crucial
Data sets written in 1995 must be readable in 2025
Support for automatic class schema evolution is a must
ROOT I/O

• The ROOT I/O subsystem is the main reason of the success of ROOT in HEP.
• It is able to stream any complex C++ objects and read them back even if the class design has changed.
• It is able to store efficiently (splitting and compression) event collections and read a subset of these collections (more than 200 PB today)
• The main component (TTree) provides a query and visualisation system.
Memory $\longleftrightarrow$ Tree

Each Node is a branch in the Tree

Queries to the data base may be on the full entry or only a subset
Large set of mathematical libraries and tools needed for event reconstruction, simulation and statistical data analysis.
Statistical Software Tools

RooFit/RooStats
Estimate parameters of an hypothetical distribution from the observed data distribution.

- \( y = f(\, x \mid \theta \,) \) is the fit model function.

Find the best estimate of the parameters \( \theta \) assuming \( f(\, x \mid \theta \,) \).

**Example**

Higgs \( \rightarrow \gamma\gamma \) spectrum

We can fit for:
- the expected number of Higgs events
- the Higgs mass
Common interface class (**ROOT::Math::Minimizer**)  
Existing implementations available as plug-ins:  
- **Minuit** (based on class `TMinuit`, direct translation from Fortran code)  
  - with Migrad, Simplex, Minimize algorithms  
- **Minuit2** (new C++ implementation with OO design)  
  - with Migrad, Simplex, Minimize and Fumili2  
- **Fumili** (only for least-square or log-likelihood minimizations)  
- **GSLMultiMin**: conjugate gradient minimization algorithm from GSL (Fletcher-Reeves, BFGS)  
- **GSLMultiFit**: Levenberg-Marquardt (for minimizing least square functions) from GSL  
- **Linear**: for least square functions (direct solution, non-iterative method)  
- **GSLSimAn**: Simulated Annealing from GSL  
- **Genetic**: based on a genetic algorithm implemented in TMVA  

All these are available for ROOT fitting and in **RooFit/RooStats**  
Possible to combine them (e.g. use Minuit and Genetic)  
Easy to extend and add new implementations
RooStats Project

- Collaborative project to provide and consolidate advanced statistical tools needed by LHC experiments
- Joint contribution from ATLAS, CMS, ROOT and RooFit
  - developments over-sighted by ATLAS and CMS statistics committees
  - initiated from previous code developed in ATLAS and CMS
  - used by both collaborations
RooStats Goal

- Common framework for statistical calculations
- Work on arbitrary models and datasets
  - Factorize modeling from statistical calculations
- Implement most accepted techniques
  - Frequentists, Bayesian and likelihood based tools
- Possible to easily compare different statistical methods
- Provide utility for combinations of results
- Using same tools across experiments facilitates the combinations of results
**RooStats Technology**

- **Built on top of RooFit**
  - generic and convenient description of models (probability density function or likelihood functions)
  - provides *workspace* (RooWorkspace)
    - container for model and data and can be written to disk
    - inputs to all RooStats statistical tools
    - convenient for sharing models (e.g. digital publishing of results)
  - easily generation of models (workspace factory and HistFactory tool)
  - tools for combinations of model (e.g. simultaneous pdf)

- **Use of ROOT core libraries:**
  - minimization (e.g. Minuit), numerical integration, etc...
  - additional tools provided when needed (e.g. Markov-Chain MC)

---

R.Brun: ROOT: data storage & analysis framework

May 22 2015
RooStats Design

- C++ interfaces and classes mapping to real statistical concepts

Diagram:

- IntervalCalculator
- HypoTestCalculator
- CombinedCalculator
- ProfileLikelihoodCalculator
- HypoTestResult
- HybridCalculator
- FrequentistCalculator
- AsymptoticCalculator
- BayesianCalculator
- FeldmanCousins
- MCMCInterval
- MCMCCalculator
- PointSetInterval
- SimpleInterval
- HypoTestInverter
- GetInterval
- GetHypoTest
- ConfidenceInterval
- Use relationships between classes
RooStats Calculator classes

**Interval Calculators**

- ProfileLikelihoodCalculator
  - interval estimation using asymptotic properties of the likelihood function
- BayesianCalculator
  - interval estimation based on Bayes theorem using adaptive numerical integration
- MCMCCalculator
  - Bayesian calculator using Markov-Chain Monte Carlo

**HypoTest Calculators**

- HypoTestInverter
  - invert hypothesis test results to estimate an interval
    - CLs limits, FC interval
- NeymanConstruction and FeldmanCousins
  - frequentist interval calculators

- HybridCalculator, FrequentistCalculator
  - frequentist hypothesis test calculators using toy data (difference in treatment of nuisance parameters)
- AsymptoticCalculator
  - hypothesis tests using asymptotic properties of likelihood function
The Toolkit for Multivariate Data Analysis provides a machine learning environment for the processing and parallel evaluation of sophisticated multivariate classification techniques. The package includes:

- Rectangular cut optimization
- Projective likelihood estimation (PDE approach)
- Multidimensional probability density estimation (PDE - range-search approach)
- Multidimensional k-nearest neighbor classifier
- Linear discriminant analysis (H-Matrix and Fisher discriminants)
- Function discriminant analysis (FDA)
- Artificial neural networks (three different implementations)
- Boosted/Bagged decision trees
- Predictive learning via rule ensembles (RuleFit)
- Support Vector Machine (SVM)
Using R in ROOT with the ROOT-R package

Lorenzo Moneta (CERN),
Omar A. Zapata M. (GSOC student)
C++ Interface to R

- ROOT-R makes use of the R packages Rcpp and RInside
  - packages providing integration of R with C++
    - mapping back and forth of R object to C++ classes
    - calling R commands from C++ code
  - ROOT-R provides an additional extra layer to Rcpp and RInside
    - simple way to call R functions from ROOT prompt or C++ code
    - facilitate conversion between ROOT/C++ objects and R objects
- Plug-ins can be developed to hide detail of ROOT-R interface
  - ROOT Minimizer plug-in using optimisation packages from R (RMinimizer)
References

- ROOT-R code
  - available now as a github branch
  - will be soon added in central ROOT git master (6.03.05)
  - with example tutorials in tutorials/r directory

- ROOT-R User Guide

- ROOT-R Web page
  - http://root.cern.ch/drupal/content/how-use-r-root-root-r-interface
Many other interfaces

- XROOTD, Dcache
- clouds services, globus, http, kerberos, ldap, bonjour
- GSL, Vc, Unuran, fftw, vdt, foam
- OpenMP, MPI
- Oracle, pgsql, sapdb, mysql, hdfs, chirp, xml, etc
- Mathematica, LabView
- Graphviz, Qt
- Pythia, Herwig, Geant4
Summary

- Continuous development since 1995
- ROOT available on most OS
- 11 full time developers at CERN, 1 at FNAL
- > 300 people have contributed at least one class
- 8500 registered users at the Forum
- 15,000 downloads per month
- I/O and Statistics are key areas
You might have heard about the Higgs Machine learning challenge (HiggsML https://higgsml.lal.in2p3.fr) that has been organized last year. Official simulated ATLAS events were publicly released, and participants competed to invent the most powerful algorithms to improve the statistical significance of the Higgs to tau +tau- signal.

The participation was overwhelming with more than 1700 teams, Machine Learning specialists, physicists and students, submitted more than 30,000 solutions, making it the most popular www.Kaggle.com challenge at the time. During these few months, a typical HEP problem has been tackled with the most advanced machine learning techniques, which have quickly outperformed traditional HEP tools (TMVA in particular, but improvements to TMVA triggered by the challenge are already in the pipeline).

This mini workshop is a step towards importing the lessons of the challenge back into High Energy Physics.

The winners will be visiting CERN Tuesday 19th May and will participate to a special (webcasted) workshop at 3PM in the CERN Main Auditorium, see http://cern.ch/higgsml-visit.
Add-on slides
Objectivity

- Storing data in a **central data base** at a time of the first discussions about **grids**.
- Objects in memory simply dumped to disk with a **primitive non-scaling system** (64 TBytes max).
- **Machine dependent format**
- **Transient format = persistent format**
- **No query mechanism !!!**
- **Storage format (object-wise)** incompatible with **member-wise access** required by analysis systems.
- **Storage format making wide-area access inefficient.**
ROOT Trees vs Objectivity

- Compared to the best candidate in 1995 (Objectivity) ROOT supports a persistent class that may be a subset of the transient class.
- ROOT supports compression (typical factors 3 to 6), file portability and access in heterogeneous networks.
- ROOT supports branch splitting that increases drastically the performance when reading.
- ROOT supports automatic class schema evolution.

R.Brun: ROOT: data storage & analysis framework  May 22 2015
ROOT Trees vs Oracle

- Same remarks as for Objectivity, in addition
- ROOT stores information *column-wise* and NOT *row-wise* like in relational data bases. This is essential when reading back very large collections that cannot use the classical data base indexing because of memory limitations.
- However relational data bases are routinely used with ROOT to store book-keeping information.
ROOT: 1995-1998

- First priority: provide a very efficient and complete replacement for PAW and ZEBRA:
  - Histograms, graphs
  - Visualization and UI
  - Minimization (Minuit) and CERNLIB replacement
  - Generic I/O with many restrictions thanks to ROOTCINT parser and Streamer functions.
  - Streamers could be customized
  - Interfaces ROOT ↔ Objectivity facilitate comparisons: functionality and performance.
- FermiLab chooses ROOT for Run2 (fall 98)