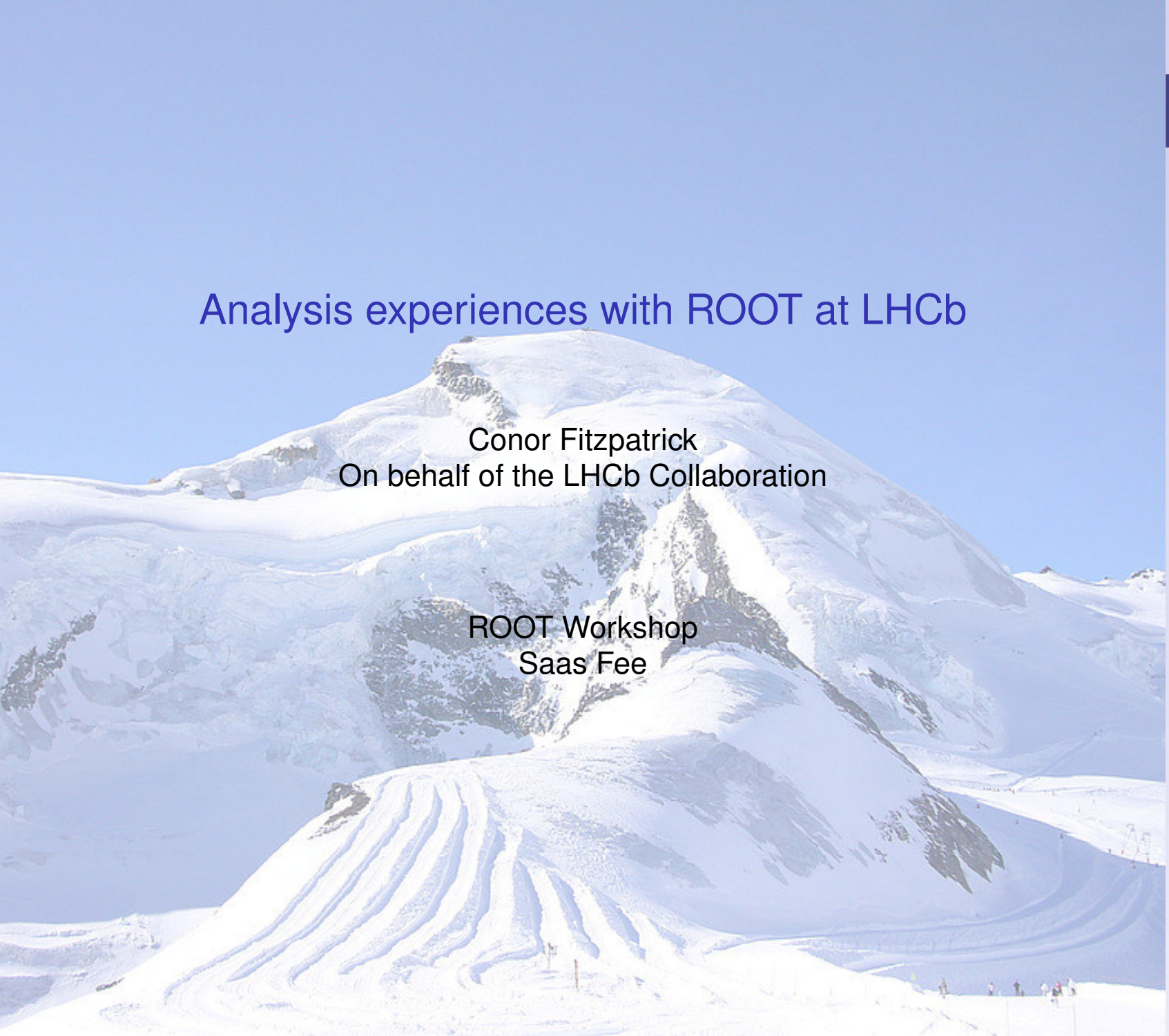


Analysis experiences with ROOT at LHCb

Conor Fitzpatrick
On behalf of the LHCb Collaboration

ROOT Workshop
Saas Fee

C. Fitzpatrick

March 13, 2013

Introduction

- LHCb is a precision beauty and charm experiment at LHC
- Precision measurements are compute and I/O intensive:
  - large datasets
  - statistical tests and systematic studies with toy monte-carlo
  - complicated fits
- ROOT is an integral component of every analysis at LHCb
- This talk will attempt to describe how analysts at LHCb use ROOT

C. Fitzpatrick

March 13, 2013

A typical LHCb analysis

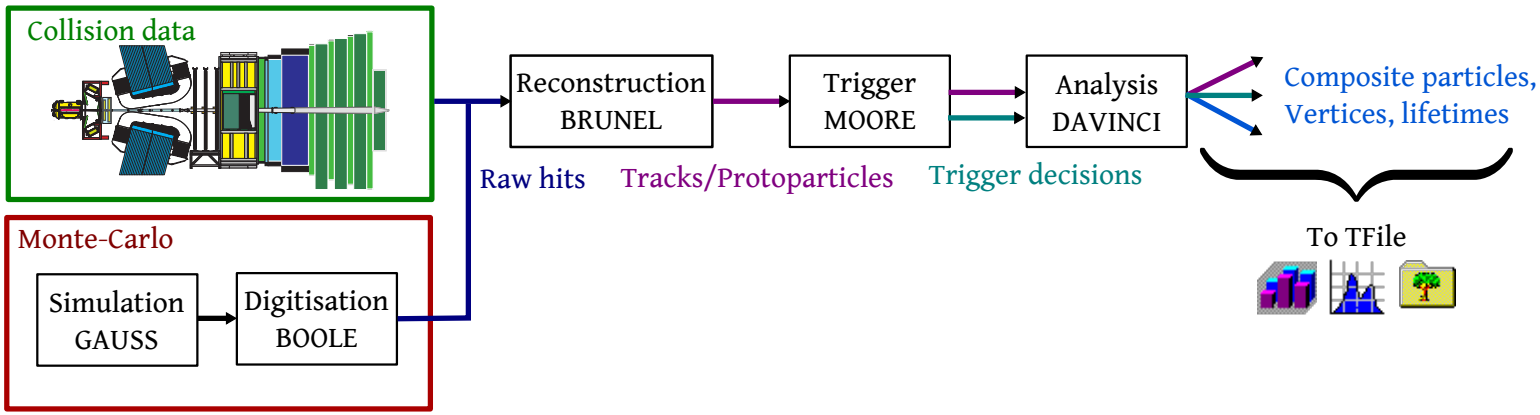
- This title is an oxymoron- no analysis is typical.
- There are as many ways to obtain and analyse data at LHCb as there are analysts.
- There are a few common themes however:
- Generally LHCb deals with precision measurements of parameters accessible though known decays, eg:
  - "Cut-and-count" direct-$\mathcal{CP}$ measurements
  - Time-dependent $\not{\mathcal{CP}}$ measurements
  - Rare decay searches
- In nearly all cases the final results are obtained from studying an offline `TNtuple...`

C. Fitzpatrick

March 13, 2013

A typical LHCb analysis: Getting to ntuple

- The LHCb software consists of several projects built on the `GAUDI` framework
- GAUDI depends on ROOT (See Ben's talk this afternoon), allowing analysts to make use of ROOT objects
- Each project serves a specific task:

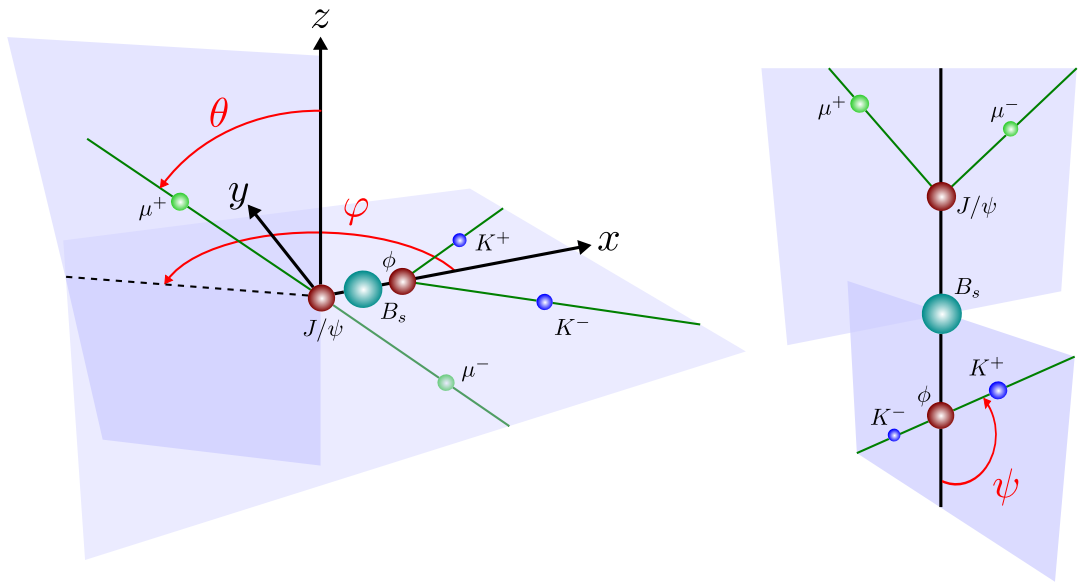


- Analysts use DaVinci to create composite particle decays, apply selections and choose what quantities they write to ntuple
- DaVinci allows users to run arbitrarily complicated algorithms on reduced/skimmed/stripped datasets, eg: TMVA-based selections are common
- Most of the 'hard work' is done here reducing event information into a few specific properties of selected particle decays
- This means the offline ntuple is very simple. Typical information stored in the ntuple:
  - 4-vectors, masses, vertex positions + errors, fitted lifetimes etc saved as float branches for each particle in the decay

C. Fitzpatrick

March 13, 2013

ϕ_s in $B_s^0 \to J/\psi\phi$

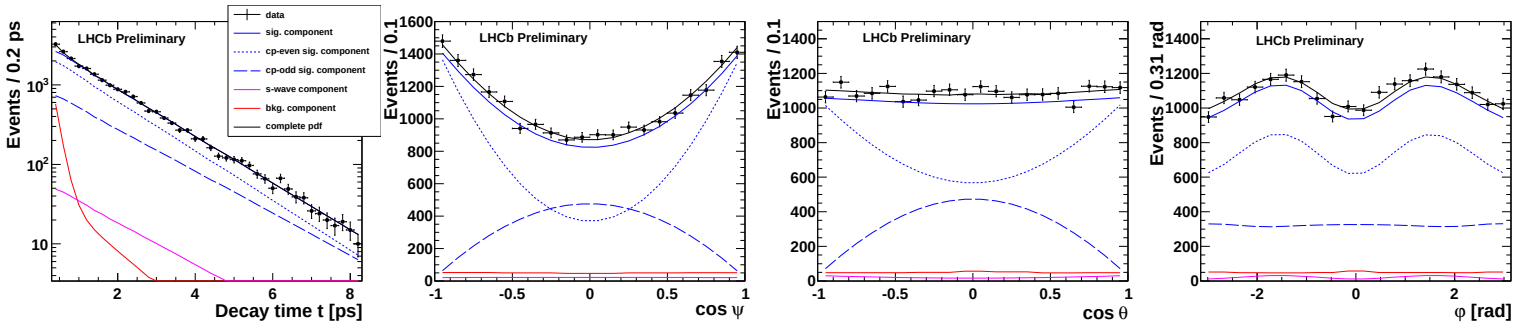


- One of the flagship analyses at LHCb [LHCb-CONF-2012-002]
- ϕ_s is a $\mathcal{CP}$-violating phase extracted from a fit to the lifetime of B_s^0 mesons to the $J/\psi\{\mu^-\mu^+\}\phi\{K^+K^-\}$ final state.
- simultaneous, multidimensional fit to three angles, lifetime over 3 tagging categories and different tagger and trigger types
- Approx 28000 signal candidates in $1fb^{-1}$of 2011 LHCb data, but toy studies run up into the millions
- Ntuple contents: B_s^0 mass, ϕ mass, lifetime, flavor tag, per-event mistags for several tagging categories, 3x angles

C. Fitzpatrick

March 13, 2013

ϕ_s fitting



- Three groups work on the ϕ_s analysis, each using different fitting technologies:
  - Two groups write their own compiled C++ fitters that interface to MINUIT and make heavy use of ROOT.
  - One group uses RooFit with a custom-written PDF.
- The three groups operate independently and allow for a powerful cross-check of the code implementation.
- In the end, all three groups agree across all the physics parameters at the permille level.
- I'll focus on one particular fitter and their experiences with ROOT:

C. Fitzpatrick

March 13, 2013

RapidFit

- RapidFit started out as a way to perform several classes of similar fit in a straightforward manner:
  - C++ binary compiled against ROOT for local use, ROOT library used for grid submission: Run anywhere ROOT is installed.
  - XML control files to steer the fit containing Parameter ranges, names, fit type, etc
  - PyROOT used to pass runtime arguments when sent to the grid.
  - Simple classes containing useful, speed-optimised functions commonly used in a PDF: Commonly used integrals, etc
- Main libraries used from ROOT:
  - `TFoam` used heavily for toy dataset generation
  - `TMinuit`, `TMinuit2` for the minimisation
  - `Math::AdaptiveIntegratorMultiDim`, `Math::IntegratorOneDim` for normalisation
  - `TTree` for input data and to store toys
  - `TH1`, `TGraph`, `TGraphErrors` to plot results
- a handful of other `Math::` functions used as well.

C. Fitzpatrick

March 13, 2013

Profiling/Optimisation

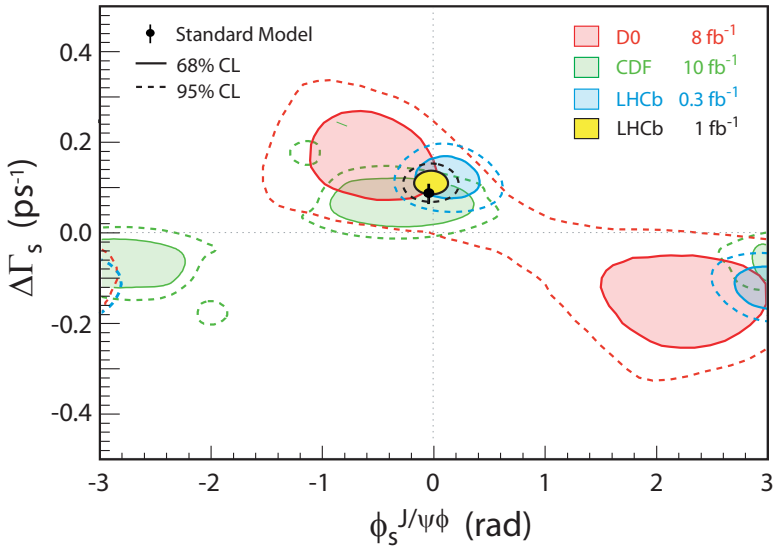
- RapidFit was extensivley profiled and optimised:
- Single fit to data to get central value takes approx 8 hours on a single core
  - 28000 datapoints: PDF evaluated once per MINUIT call per datapoint
  - Several calls to normalisation method required when acceptance functions are included.
  - PDF is cached extensively as little changes within each call.
  - 600-1000 Minuit calls per fit
  - 5×10^7 calls to the PDF normalise/evaluate methods for a single fit
- pthreads now used for single fit, speedup is 8 on an 8-core machine. Lack of thread-safety in the original code required heavy rewriting

C. Fitzpatrick

March 13, 2013

Feldman-Cousins

- Rapdfit is also used to make Feldman-Cousins 1D and 2D contour scans for the ϕ_s analysis:
  - thousands of toy datasets generated of equivalent size to the real data at thousands of discrete points on a 1D or 2D plane
  - Each toy dataset fit to twice
  - Confidence belts determined from likelihood ratio of toys at each datapoint.
- This is extremely compute intensive, but "Embarassingly parallel":
  - Toys generated at one point in phase space are independent of those elsewhere, so jobs can be divided up and sent to the grid
  - Toy fit results stored and saved in ntuple format, merged and processed locally later.
- The full FC scan for the last publication consumed about 2.5 CPU years on the grid.



C. Fitzpatrick

March 13, 2013

RapidFit summary

- The analysts who worked on RapidFit found a lot of the built-in ROOT functionality invaluable:
  - I/O: The speed, portability and merging functionality of ROOT's data formats
  - TFoam and integrator methods were fast and accurate
  - Ability to ship Rapidfit as a ROOT library made grid deployment straightforward
- They encountered one or two issues along the way too:
  - Memory leak checking turned up a lot of issues "upstream" in the ROOT libraries
  - The analysts were unaware of the existence of PROOF or PROOF-lite and went the way of pthreads
  - Persistency with `TFoam` caused some problems: Calling delete didn't free memory, re-using an existing instance required heavy code modification
  - Customising plots took a significant amount of time

C. Fitzpatrick

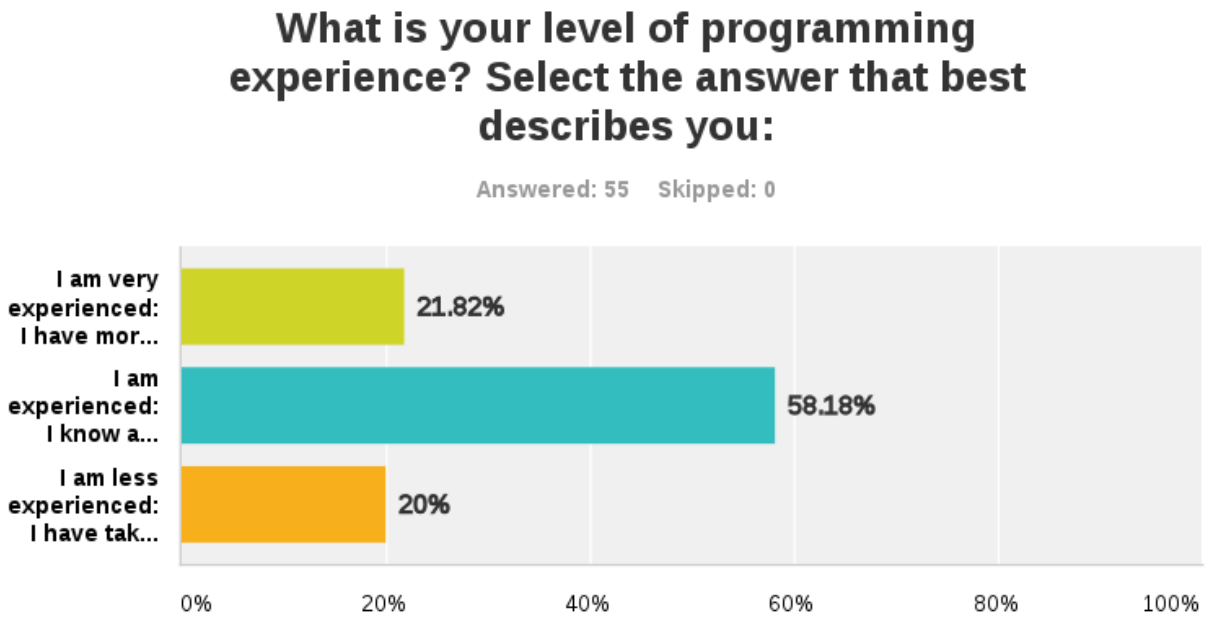
The LHCb ROOT User survey

- Hopefully the last slides gave you an idea of how ROOT was used in a specific analysis
- To get a broader feel I've conducted a surveymonkey poll of LHCb analysts
- I got 55 respondents, which covers a reasonable slice of the collaboration involved in analysis tasks
- The survey consists of 10 questions, of which the earlier ones gauge experience and the last two are open comments
- I've loosely summarised the comments by picking out common themes

C. Fitzpatrick

March 13, 2013

Question 1: Experience

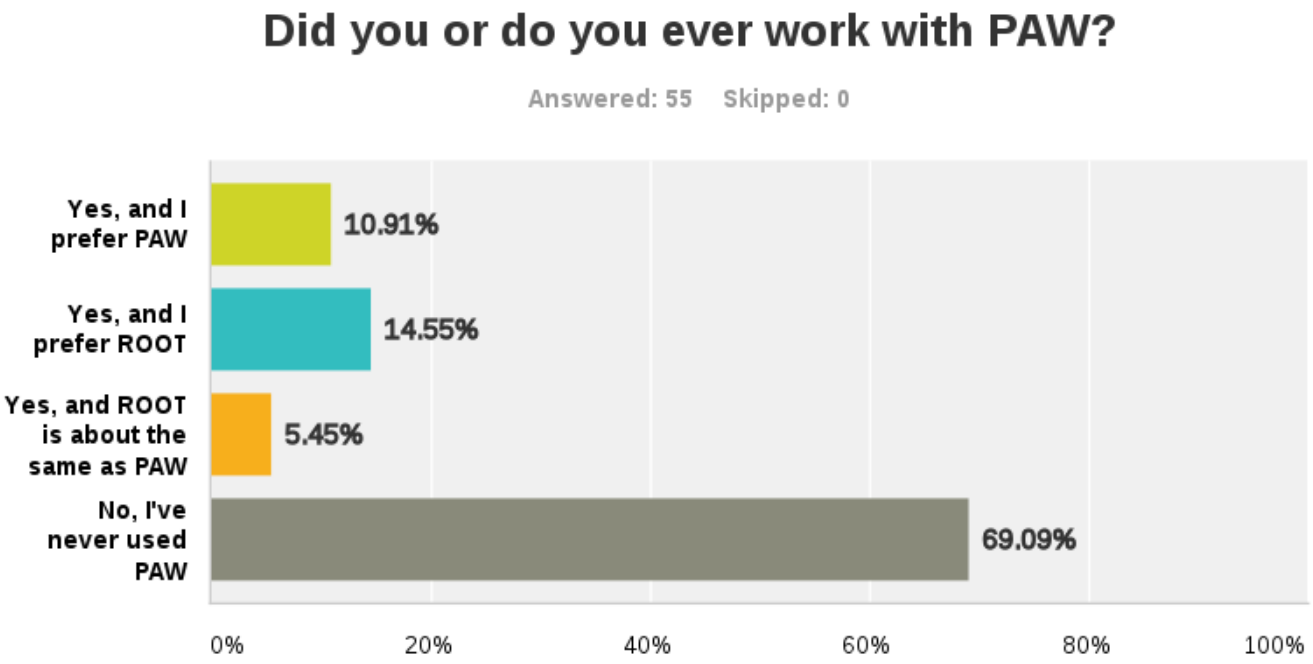


Answer Choices	Responses	
I am very experienced: I have more than 5 years experience in at least 3 programming languages, I know what Object-Oriented code is, and how to write it. I profile my code. I write code that can be reused by myself and others easily, and I keep it maintained.	21.82%	12
I am experienced: I know a couple of programming languages well, I know what object oriented code is. I frequently re-use my own code with minor modifications.	58.18%	32
I am less experienced: I have taken some basic programming courses, but most of my understanding comes from ROOT tutorials. I write code/scripts to get results and copy and paste code snippets in order to reuse them.	20%	11
Total		55

C. Fitzpatrick

March 13, 2013

Question 2: PAW

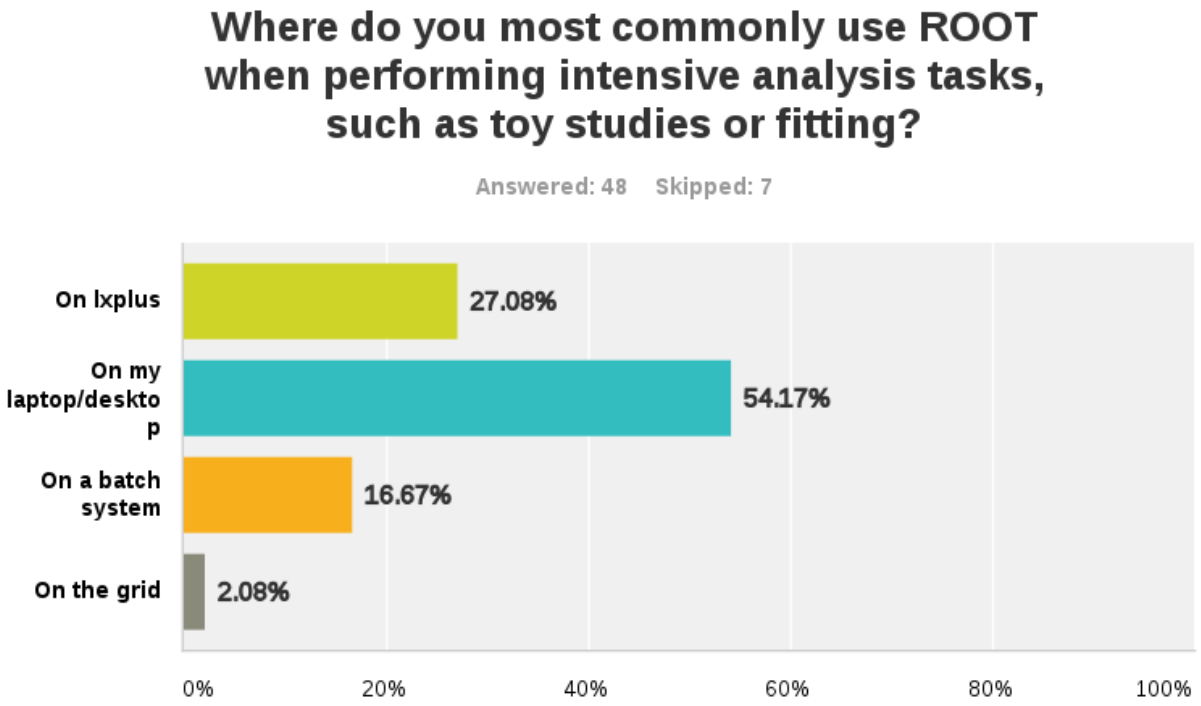


Answer Choices	Responses	
Yes, and I prefer PAW	**10.91%**	6
Yes, and I prefer ROOT	**14.55%**	8
Yes, and ROOT is about the same as PAW	**5.45%**	3
No, I've never used PAW	**69.09%**	38
Total		55

C. Fitzpatrick

March 13, 2013

Question 3: Hardware



Answer Choices	Responses	
On lxplus	27.08%	13
On my laptop/desktop	54.17%	26
On a batch system	16.67%	8
On the grid	2.08%	1
Total		48

C. Fitzpatrick

March 13, 2013

Question 4: Usage preference

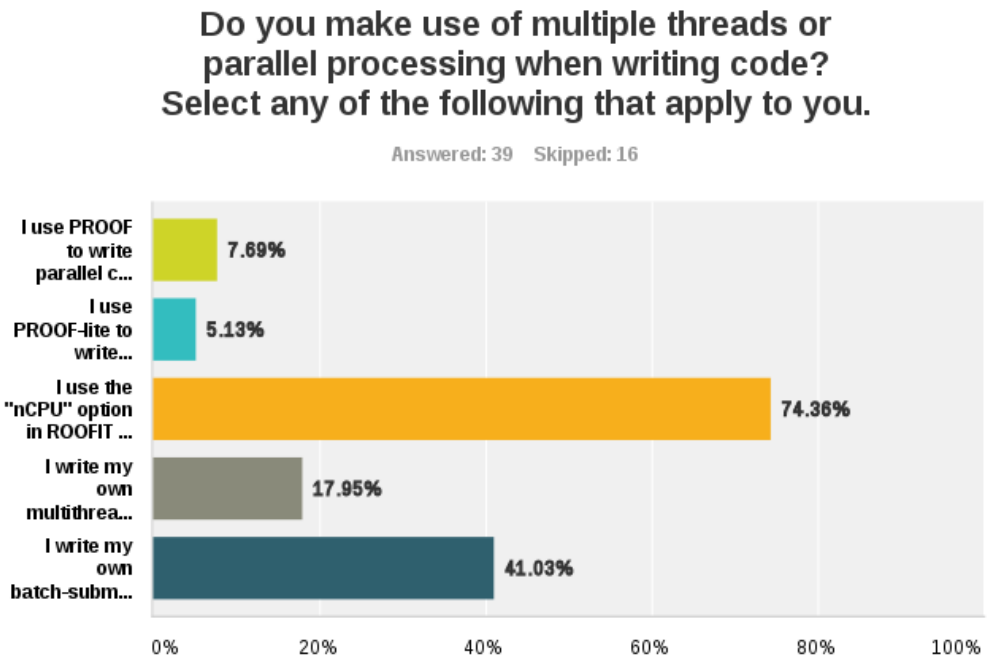
What ways do you commonly interact with ROOT? Rank these options according to highest usage:

	Rank				
Usage	1	2	3	4	5
Compiling C++ binaries	39.62%	18.87%	26.42%	9.43%	20.75%
Using the TBrowser	5.66%	35.85%	26.42%	18.87%	13.21%
Writing ROOT macros	24.53%	18.87%	26.42%	9.43%	20.75%
Using CINT	5.66%	13.21%	22.64%	43.40%	15.09%
Writing PyROOT scripts	24.53%	13.21%	15.09%	9.43%	37.74%

- Interpretation of this table is tricky, as correlations aren't included
- Generally, LHCb users prefer to compile their analysis code to get better debugging information from GCC
- While PyROOT comes last, it's the more experienced analysts that tend to use it

C. Fitzpatrick

Question 5: Parallelism

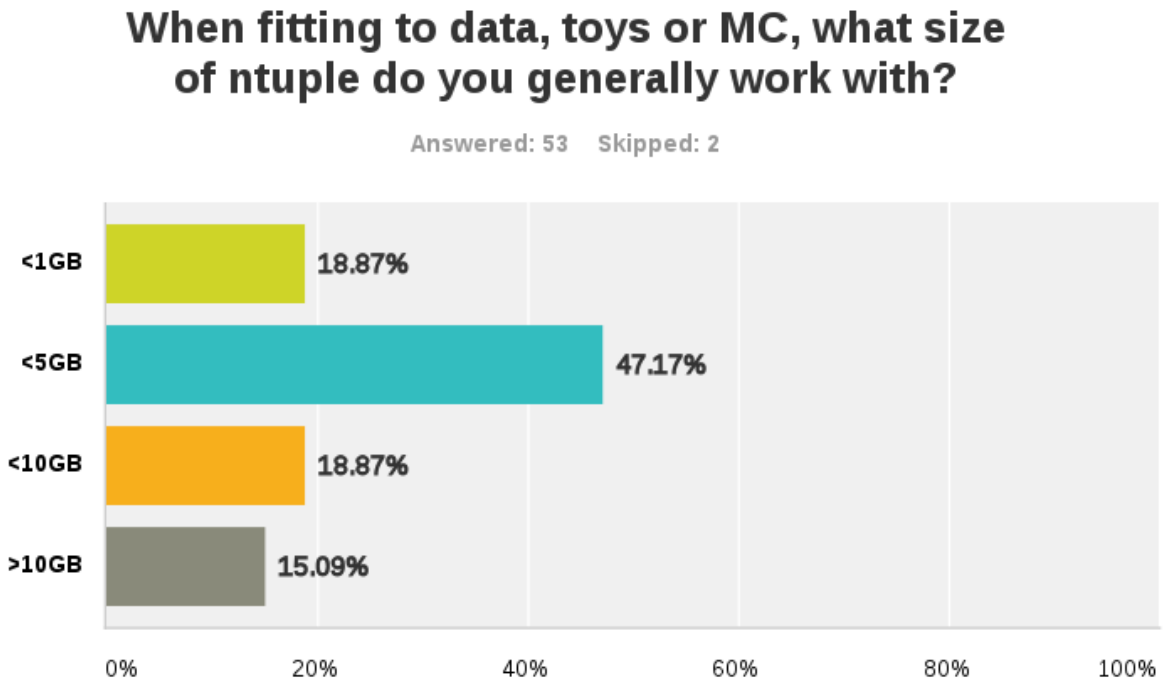


Answer Choices	Responses	
I use PROOF to write parallel code for batch submission	7.69%	3
I use PROOF-lite to write multithreaded code to run on my multi-core desktop/laptop	5.13%	2
I use the "nCPU" option in ROOFIT to speed up my fits	74.36%	29
I write my own multithreaded desktop/laptop code without the use of PROOF/PROOF-lite/nCPU	17.95%	7
I write my own batch-submitted, single-threaded analysis code and combine the results myself	41.03%	16
Total Respondents: 39		

C. Fitzpatrick

March 13, 2013

Question 6: Dataset size

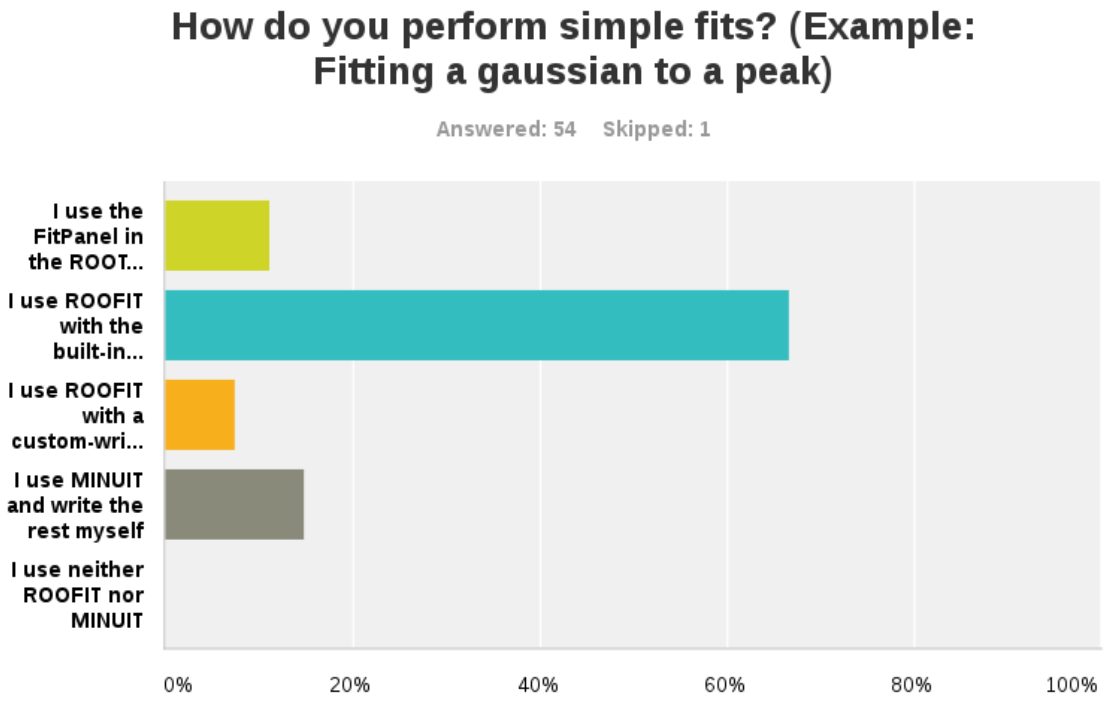


Answer Choices	Responses	
<1GB	**18.87%**	10
<5GB	**47.17%**	25
<10GB	**18.87%**	10
>10GB	**15.09%**	8
Total		53

C. Fitzpatrick

March 13, 2013

Question 7: Basic fits

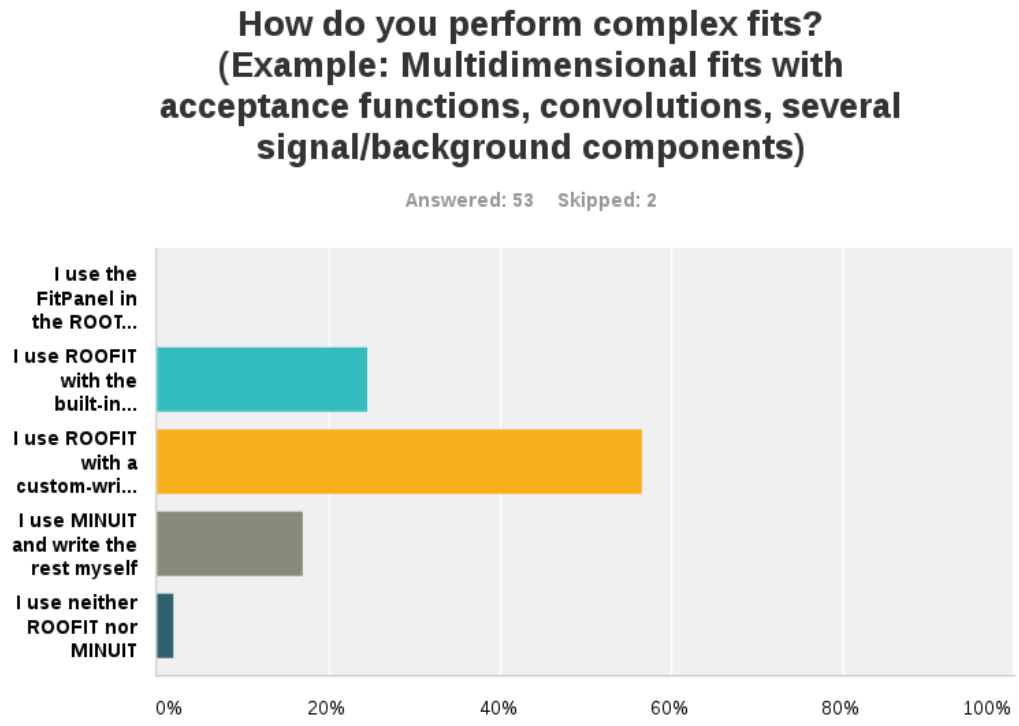


Answer Choices	Responses	
I use the FitPanel in the ROOT TBrowser	11.11%	6
I use ROOFIT with the built-in basic PDFs	66.67%	36
I use ROOFIT with a custom-written PDF	7.41%	4
I use MINUIT and write the rest myself	14.81%	8
I use neither ROOFIT nor MINUIT	0%	0
Total		54

C. Fitzpatrick

March 13, 2013

Question 8: Complicated fits



Answer Choices	Responses	
I use the FitPanel in the ROOT TBrowser	0%	0
I use ROOFIT with the built-in basic PDFs	24.53%	13
I use ROOFIT with a custom-written PDF	56.60%	30
I use MINUIT and write the rest myself	16.98%	9
I use neither ROOFIT nor MINUIT	1.89%	1
Total		53

C. Fitzpatrick

March 13, 2013

Question 9: Dislikes

What do you dislike about ROOT? Please list any problems or issues you've encountered.

Common responses:

- Inheritance structure/OO is unnatural, eg: TH1 has 3 dimensional methods
- Overuse of Run-time type information and use of raw pointers makes debugging difficult
- Objects require unique names, lots of stuff happens "under the hood" with global pointers killing encapsulation
- Passing options as strings causes many hard to locate bugs
- CINT prevents easy debugging/profiling, often randomly (and recursively!) segfaults, isn't fully compliant
- Limited STL integration (although ROOT predates STL and support has improved!).
- Lack of formatting simplicity: No auto placement of labels/legends, lots of work to make plots suitable for journals.

C. Fitzpatrick

Question 10: Likes

What do you like about ROOT? Please list anything that you find particularly useful.

Common responses:

- Fully populated and feature-rich classes/libraries eg: Math, TLorentzVector, TRandom3 etc
- RooFit, TMVA: We are glad to see these bundled and well supported
- Efficient I/O and straightforward Tree/Ntuple access eg: `TTree::Draw("variable","cuts")`
- PyROOT is loved (by those who know it)
- Helpful and generally well documented class reference
- Wide range of tutorials (though we'd like to see more of them in compilable C++)

C. Fitzpatrick

March 13, 2013

Conclusions

- LHCb uses ROOT for all analyses
- There are many ways to do the same thing with ROOT, and every analyst will have a different preference
  - Compiled C++ binaries are preferred to CINT macros, but more people are moving to PyROOT
  - The majority of analysis tasks are performed on laptops/desktops
  - ROOT's exceptional I/O performance is increasingly important at the analysis level: Datasets are ever-expanding
  - Not many analysts are aware of PROOF-lite, although many have multi-core laptops
- LHCb's analysis 'wish-list' for the future of ROOT:
  - We would like to see continued/enhanced support for ROOT-based applications, particularly ROOFIT
  - Continued PyROOT integration and support is encouraged
  - The Gaudi framework will take time to incorporate ROOT 6. We would like to see backports of bugfixes and continued support of ROOT 5
  - We would appreciate simpler, more automated formatting tools for plots
  - We find the inheritance structure, use of globals and some OO decisions in ROOT to be unintuitive
- We look forward to the many improvements and additions in ROOT 6!

C. Fitzpatrick

March 13, 2013