2015 ROOT Users' Survey

Summary of users' experience, needs and suggestions

John Harvey CERN/PH/SFT

ROOT Users' Workshop, Saas Fee, Sept 15-18th

The Survey

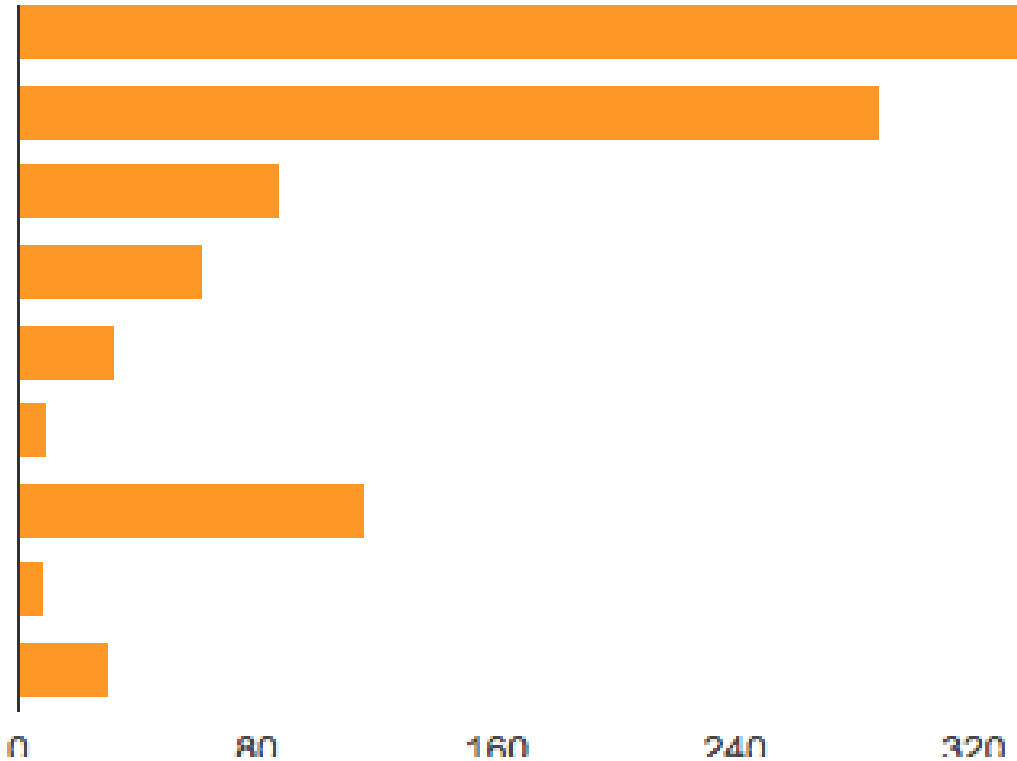
- The questionnaire
  - Background of users
  - Communication within community
  - How ROOT is used
  - Learning to use ROOT
  - Areas you would like to see improved
- The response is extremely valuable
  - here I summarise the main messages
    - see the backup slides for a more comprehensive summary
  - developers will try to address concerns in their talks
  - it will help drive the future programme of work

Profile of respondants

- 353 people filled in the questionnaire – thank you!
- Experience: <2y: 4% 2-5y: 28% 5-10y: 37% >10y: 30%
- Frequency: every day 73%; several times/week 25%
- Essentially all HEP research programmes are represented
  - LHC, ν physics, b physics, …
- Nuclear and plasma physics
  - RHIC, GSI,…
- Astronomy and astrophysics
  - Fermi-LAT, Fermi-HAWC, HESS, MAGIC, POLAR, CDMS …..

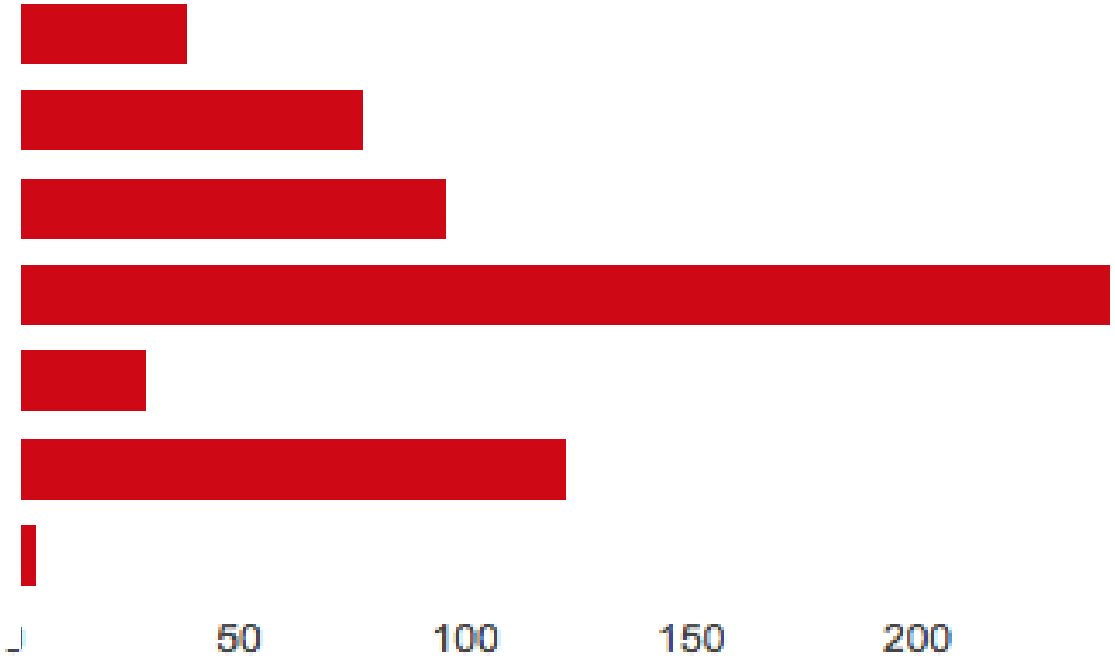
Which languages do you know?

C++	**346**	100%
Python	**289**	83.5%
Java	**88**	25.4%
JavaScript	**62**	17.9%
R	**33**	9.5%
Ruby	**10**	2.9%
Fortran	**116**	33.5%
Go	**9**	2.6%
Other	**30**	8.7%



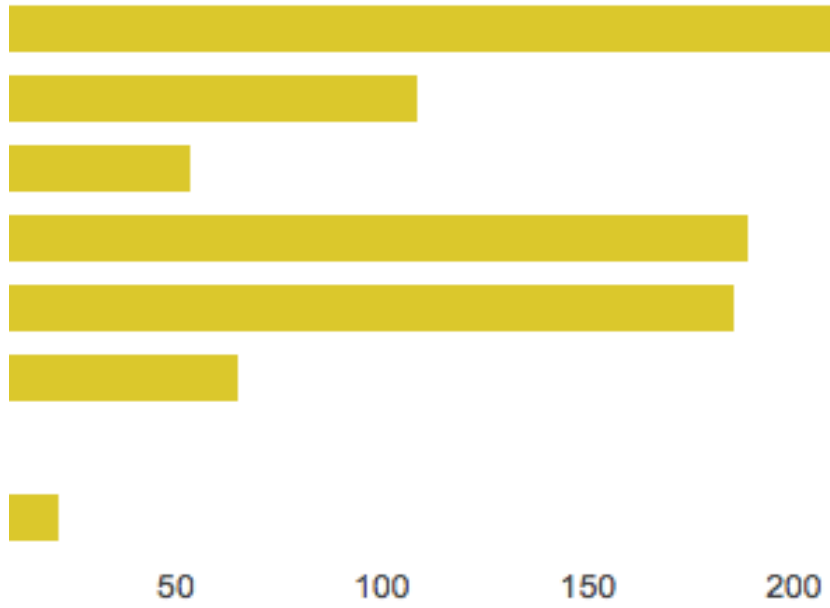
Which ROOT version do you use?

Master (head)	39	11.4%
6.04	78	22.8%
6.02	96	28.1%
5.34	243	71.1%
5.32	30	8.8%
Expt. provided	123	36%
Other	5	1.5%

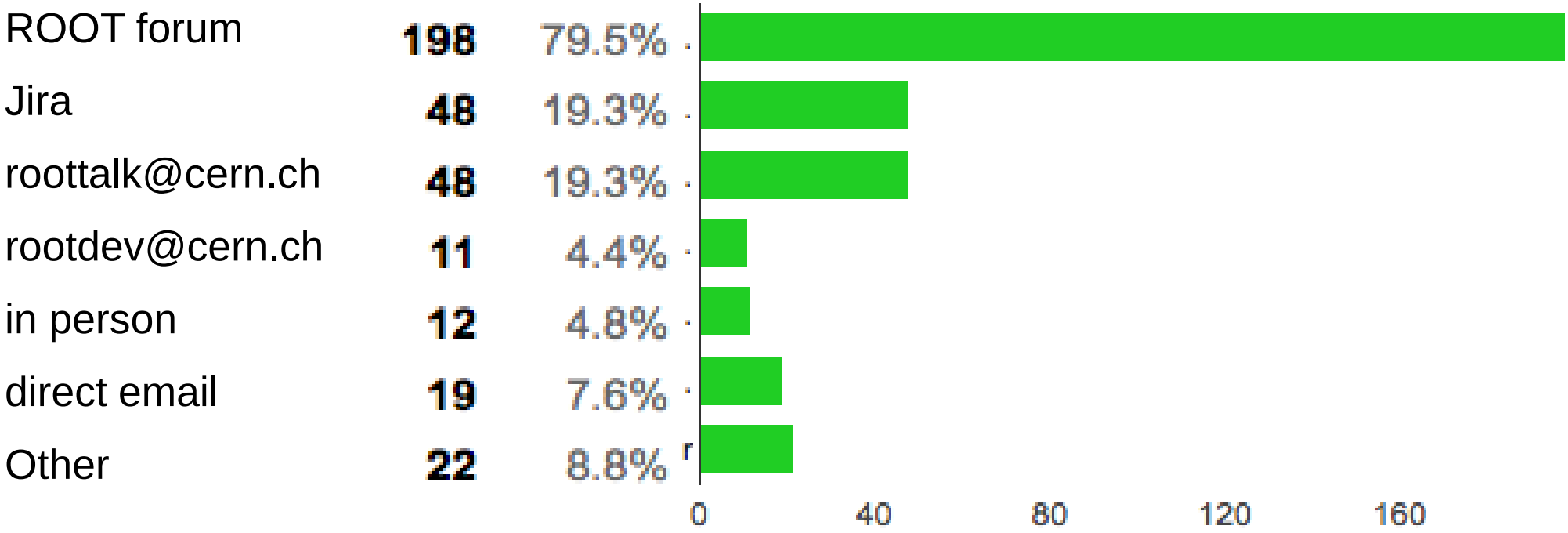


Which platforms do you like to use?

Ubuntu / Debian	**217**	62.7%
RedHat Enterprise Linux / CentOS	**109**	31.5%
Fedora	**54**	15.6%
SLC	**189**	54.6%
MacOS	**186**	53.8%
Windows	**65**	18.8%
AIX	**1**	0.3%
Other	**22**	6.4%



Communication with ROOT team



- On average 20 posts/day (4.5 new questions/day) on forum
- We use JIRA for tracking bugs
- N.B. Intention is to deprecate the roottalk mailing list

Comments on support

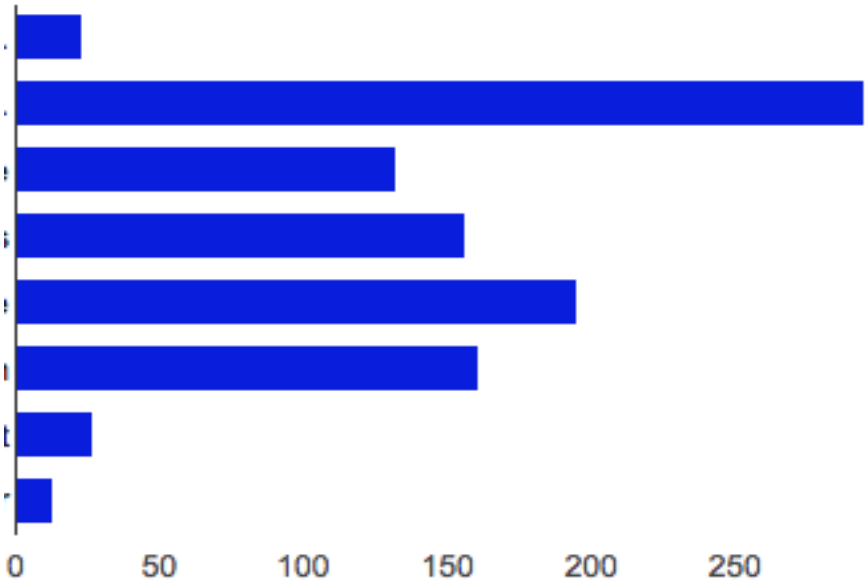
- Most users are either fully (45%) or partially(45%) satisfied
  - Usually I find answer to my problem on forum
  - I use forum a lot but almost never need to post questions
- BUT 5-10% are clearly not happy
  - ‘Improve the documentation’
  - ‘I wish tutorials were kept up-to-date and were more relevant to my needs’
  - ‘It can take very long from bug reports to fixes’
  - ‘Some questions left completely unanswered’
  - ‘I sometimes get “that’s the way it is – live with it”! ’

How ROOT is used

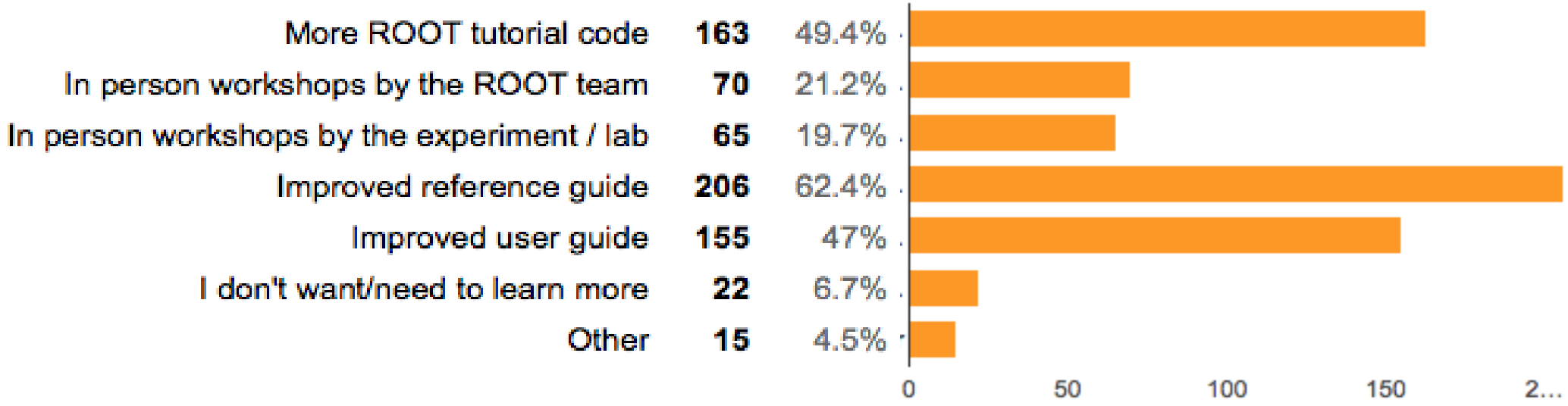
- A lot of interesting detailed information on which classes are either unknown and/or unused:
  - TRobustEstimator (99%), TRolke (99%), TH1K (95%), Recorder (100%) ….
- Useful for deciding either what needs to be made more visible, if its useful, or what should be deprecated, if its not
  - It is important that mature software (20 years old) is not allowed to grow unchecked for maintainability reasons
- A Deprecation Policy is needed :
  - announcement of the 'wish to deprecate in the next release'
  - advice to be given on recommended alternative solutions
  - allow plenty of time (>= 6 mths) for users either to adapt or to object

How do you learn more about ROOT now?

ROOT's courses	**23**	6.8%
Reference Guide (class documentation)	**295**	86.8%
User's Guide	**132**	38.8%
Colleagues	**156**	45.9%
Existing code	**195**	57.4%
Forum	**161**	47.4%
Mailing list	**27**	7.9%
Other	**13**	3.8%



How would you like to learn more?



- Tutorials need work – work is underway to exploit Notebook Technology (Jupyter)
- Reference guide – making transition to *doxygen* will help a lot
- User Guide – need a campaign to refresh it (whole team)

General comments on things to improve

- Proper compatibility with modern compiler features
  - ‘physicists can handle concepts like templates, so they shouldn't be hidden just because they can be confusing’
- Consistent naming of functionality
  - ‘e.g. why should some accessors start with *get* and others not?’
- Memory management
  - ‘many times I had to debug segmentation faults, because of unintuitive behaviour of the ROOT garbage collector’
- Better recovery from errors
  - ‘it would be good if root didn't always just blow up and show memory maps and incomprehensible stack traces ‘
- Using namespaces and getting rid of aspects due to bad use of inheritance
  - ‘e.g. it makes no sense for a TH1F to have a z axis’
- Support for multi-threading

More general comments

- ‘I would like to see it broken into many smaller libraries, so other codes could depend only on the tiny, tiny part of ROOT that they actually use’
- ‘When I need something that is missing I write it myself and incorporate it into an analysis level package’
  - need for a ‘how to contribute’ mechanism so others can benefit
- ‘No, there's way too much stuff in Root already, and plenty of other statistical packages that do the same thing. ROOT should focus on solid core functionality and interfacing with these other packages rather than reinventing the wheel.’

Specific comments on things to improve

- ‘Would greatly appreciate more stl support
  - e.g. std::string instead of TString, std::vector instead of TList’
- ‘Histogram ownership in TDirectory is confusing & error prone’
- ‘I think the machine learning field progress requires to start up a team to improve and further develop TMVA package ‘
- ‘Concerning RooFit, it would be nice to have more examples to do p0 and exclusion plots’
- ‘PyRoot: improve speed and more convenient handling of object ownership
  - better support for 2D arrays, easier way to write to TTree’
- ‘Fiddly and unpredictable behavior in plotting
  - e.g. difficult to set text sizes in a consistent way between different TPads on the same canvas. ‘
- ‘I would like more GUI functionality for doing common analysis tasks like making cuts on variables.’

Comments on documentation

- ‘I would like to see improved documentation and guides on the website - documentation is especially hard for beginners’
- ‘In particular, usage examples would help since there is very little consistency in interfaces.’
- ‘Examples of some high energy physics analysis codes and a link to a repository where they are collected’
- ‘Documentation of RooStats’
- ‘Please update Roofit manual by including the examples of RooMCStudy etc - I found its usage mainly from the forum ‘
- ‘Being able to know what is contained in each tutorial without randomly clicking until one finds an example would be excellent’

ROOT Website – this is new

- Rediscover, reorganise, modernise the original content
  - The entire content of the web was reviewed
- Priority: improve documentation
  - e.g. grouping of ROOT courses
- Creation of a "Getting Started" section
- Present topical guides and manuals in a dedicated page
- Dedicated "How to" section - planning to significantly expand it
- New and existing code examples offered in form of "Notebooks"

What do you like most about ROOT?

- Qualities : performance, robustness, flexibility, powerful
- Extremely complete and extendable
- Free, cross-platform, plotting data is simple
- Its ability to handle large amounts of data fast
- ‘The python interface is fantastic! Really Really convenient.’
- ‘The command line interpreter and tab complete functionality allow for easy C++ debugging’
- ‘Tools for all aspects of HEP are present and usable ‘
- ‘It is universally used in my work environment, so everyone's code is somewhat compatible ‘
- ‘Having moved between experiments, the transition has been eased by familiarity with ROOT allowing me to get to grips with new frameworks in a more efficient fashion’

What do you like less about ROOT?

- Comments under “to be improved” often repeated here
- ‘A confusing class structure. Inconsistencies in conventions. ‘
- ‘Error handling- “when ROOT dies it is a mess” ‘
- ‘Class design is pretty bad, maybe it's time to consider using templates’
- ‘Plotting of good looking histograms is painful ‘
  - “It can take dozens or sometimes even hundreds of lines of code to produce a reasonable plot “
- Memory management
- Namespaces
  - ‘use namespaces appropriate for the class of functions!’
- Most recurring comments concern documentation

I'll finish by quoting a specific use-case

"Once you go past the initial quick accomplishments you find out that ROOT has a sting in the tail. You realise that you need to, somehow, change the font of the axis, or ROOT is choosing strange binning, or the fitter is not stable and sometimes converges and sometimes does not. The moment that you realize you need to explore deeper into a given tools options, whether for analysis or display, you suddenly run out of documentation. Then you end up going to the code directly through the reference guide and trying to puzzle out, with very poorly commented code, what, exactly, is going on. Options are the worst, in many cases they are not listed in the code or in the User guide. And the options are often wrapped up in a different class than the one you are planning on using them in! Whether you can find that class and whether the possible options are listed is another question entirely. This is often the ROOT experience, quick success and initially satisfying results very quickly, then many many hours of searching to get exactly what you want or need as soon as you stray from the "standard" usage of a tool. I've wasted many days on some very trivial items with ROOT. "

Concluding Remarks

- Many thanks again for all your comments and for expressing them in such a comprehensive and frank way
- Please continue to provide feedback, at the workshop and after using the forum
  - developers are always happy to be contacted directly as well
- The on-going programme of work is maintained in JIRA and so is visible to all – comments are always welcome
  - it is discussed in the Architects Forum attended by representatives from the LHC experiments
- Your (new) collaborators may well want to take advantage of the training offered in the CERN training programme
  - will appear in the training catalogue soon
  - N.B. this is open to CERN Users as well as CERN Staff

Backup slides

Which areas should be improved?

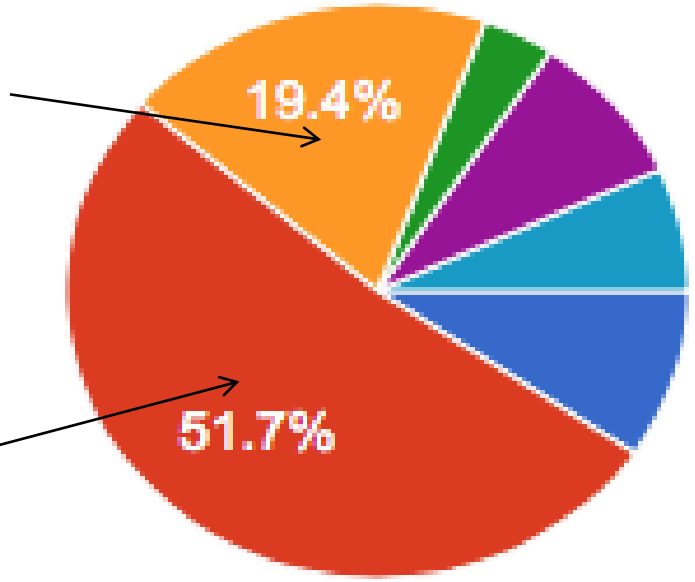
- The concept of the TTree generally feels a bit outdated to me compared to some of the other data analysis packages out there. I understand that in particle physics we work on an event by event basis, and each event can be an entry in a tree, but it would be nice to have something more like a Pandas dataframe, where you can perform simple vectorized operations that combine different columns. Given the size of our datasets, I know it's probably not practical memory wise to do something like this, but I still feel that a dataframe like structure where you can simply refer to variables via a string name rather than having to go through the usual MakeClass SetBranchAddress etc. procedure would just be more intuitive. TTree::Draw already offers some of this but I think reforming things more fundamentally would pay off big.

How did you install ROOT?

I don't know, I am using a preinstalled version	**33**	9.5%
I built ROOT from source	**179**	51.7%
I downloaded binaries / tar file / installer from the ROOT web site	**67**	19.4%
From an external distribution (MacPorts etc ...)	**13**	3.8%
used my operating system's package system (apt-get, yum, zypper, pacman etc)	**30**	8.7%
Other	**24**	6.9%

download from ROOT website

built from source



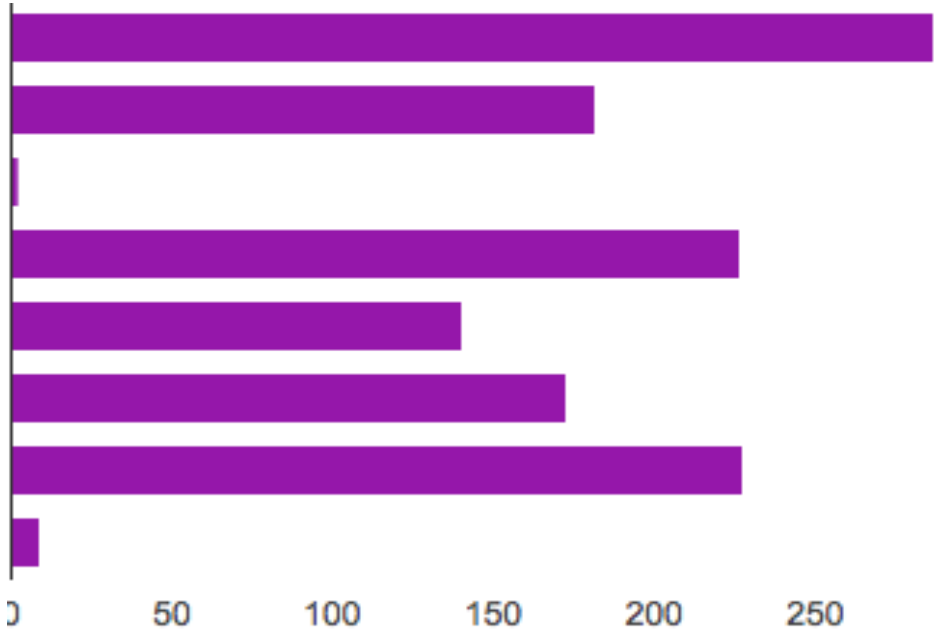
Are you happy with support?

	YES	Partially	No	Hopeless	N/A
they know what they are doing	34	21	2	2	41
the response is relevant	25	27	4	1	44
the response is quick	21	23	9	3	45
the response is by experts	32	33	2	0	4
the response is relevant	26	35	6	0	4

- nearly half of users are not using the forum or interacting with developers
- ~ one half of the rest are fully satisfied, with most of the remainder partially satisfied
- speed of response is perhaps one thing to focus on

Which Interfaces?

ROOT's C++ prompt (CINT / cling)	**287**	82.9%
ROOT's python interface	**182**	52.6%
ROOT's Ruby interface	**2**	0.6%
Building my own application	**227**	65.6%
Through the Graphical User Interface	**140**	40.5%
Through shell scripts invoking ROOT	**173**	50%
In batch mode (no graphics)	**228**	65.9%
Other	**9**	2.6%



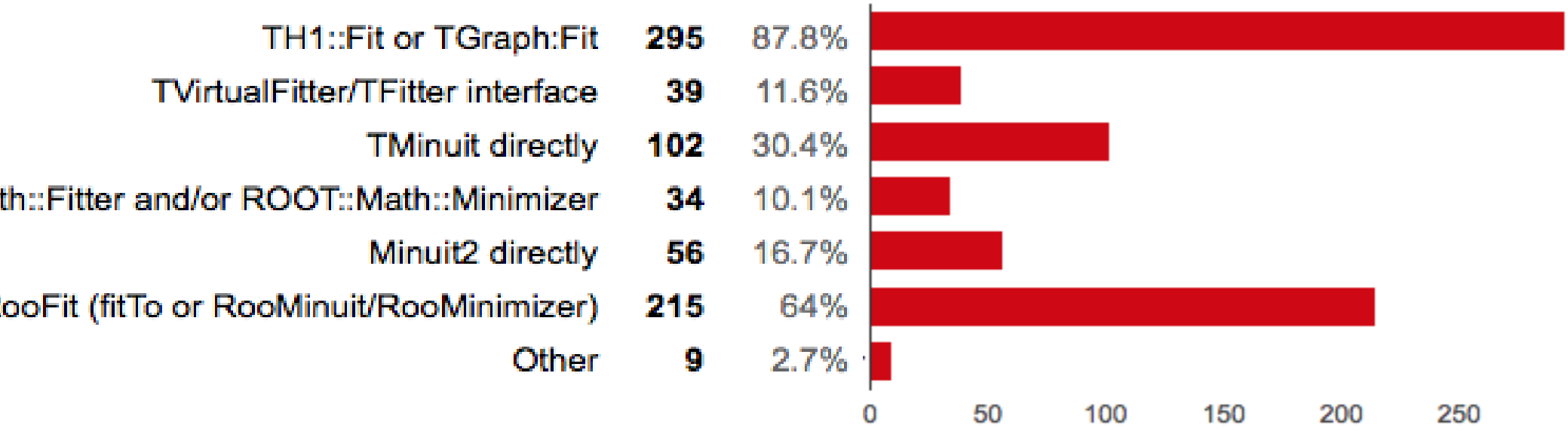
Histogramming & Data Analysis Classes

	Don't Know	Don't Use	Use
Automatic binning : histogram buffer	36	25	40
TH1K : nearest K neighbours	85	10	5
TKDE : kernel density estimators	82	16	3
RooKeysPdf: KDE in RooFit	65	18	17
TMultiDimFit:	71	25	5
TBinomialEffieciencyFitter	76	19	5
TFractionFitter	55	29	17
TGraphSmooth	65	26	9
TSplines	47	39	15
TPrincipal	83	14	3

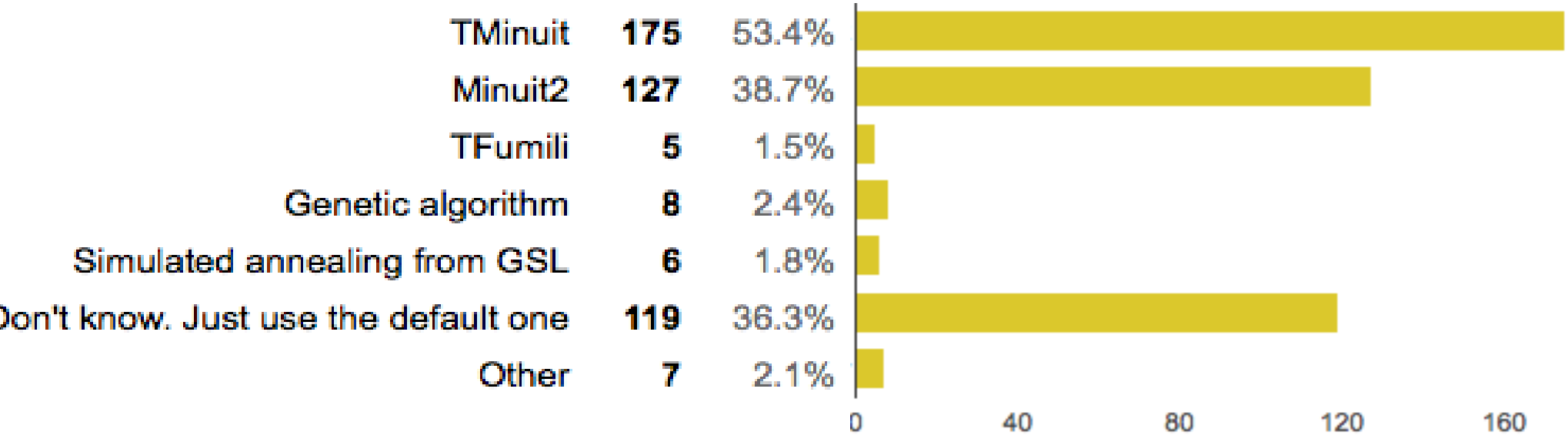
Histogramming & Data Analysis Classes

	Don't Know	Don't Use	Use
TSpectrum as a peak finder	75	18	7
TSpectrum for background subtraction	80	16	3
TRobustEstimator	91	9	1
Quadratic programming	92	7	1
TKDTree/TKDTreeBinning	92	6	1
TMultiLayerPerceptron	72	20	8
TMVA	18	33	49

Fitting interfaces



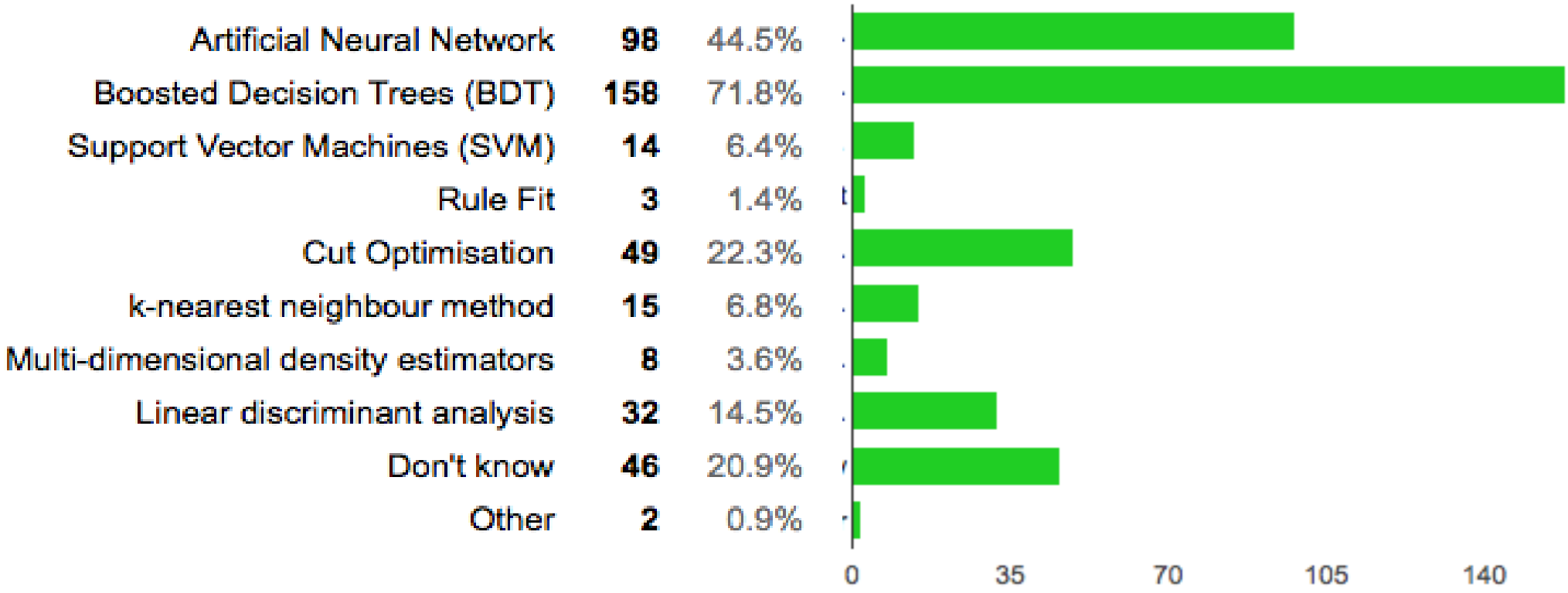
Fitting algorithms



RooFit and TMVA

	RooFit	TMVA
Its good enough	31	37
Interface is fine but not performance	6	5
Its too difficult to use	15	3
Support is not good enough	12	4
I prefer to use something else	4	2
I don’t use it	25	44
Other	8	5

Which TMVA Methods do you use?



What features do you miss in RooFit, TMVA?

- TMVA is good in general, but most algorithms train (and evaluate) so slowly that it becomes impractical for testing
- RooFit documentation – mentioned several times!
- Code examples that could be reused, a tutorial like RooStats
- Set of user guides, starting from basic and getting more sophisticated
- When I need something that is missing I write it myself and incorporate it into an analysis level package
  - e.g. data combination software like: http://blue.hepforge.org/
- No, there's way too much stuff in Root already, and plenty of other statistical packages that do the same thing. ROOT should focus on solid core functionality and interfacing with these other packages rather than reinventing the wheel.

..and there are plenty of specific suggestions…

- Deep neural networks (YAML, ..)
- Deep learning, TMVA only implements archaic techniques
- Need convenient tools for calculating p-values, limits. Existing tools either not documented well or are statistically incorrect.
- Miss tools to split likelihood minimization across machines
- TMVA: make possible different cuts for training/testing
- TMVA: Support for Cross-Validation, Grid-Search of Hyperparameters
- More flexibility in the TTree manipulation
- TFractionFitter with weighted events
- Genetic algorithms (e.g. NEAT)
- ……

….and more….

- In some ROOT installation/distribution TMVA or RooFit are not included.
- HistFactory should provide by default a possibility to use parametric modelling, ….
- RooFit: Wavelets i.e. "fourier" in both frequency & position
- Can we have variable list and editor back in the same window in TBrowser, please?
- Random forests
- TMVA: Support for Cross-Validation, Grid-Search of Hyperparameters (OptimizeAllMethods does not work properly for me)
- TCLs - should exist!
- Multi-dimensional target using Boosted Regression Tree

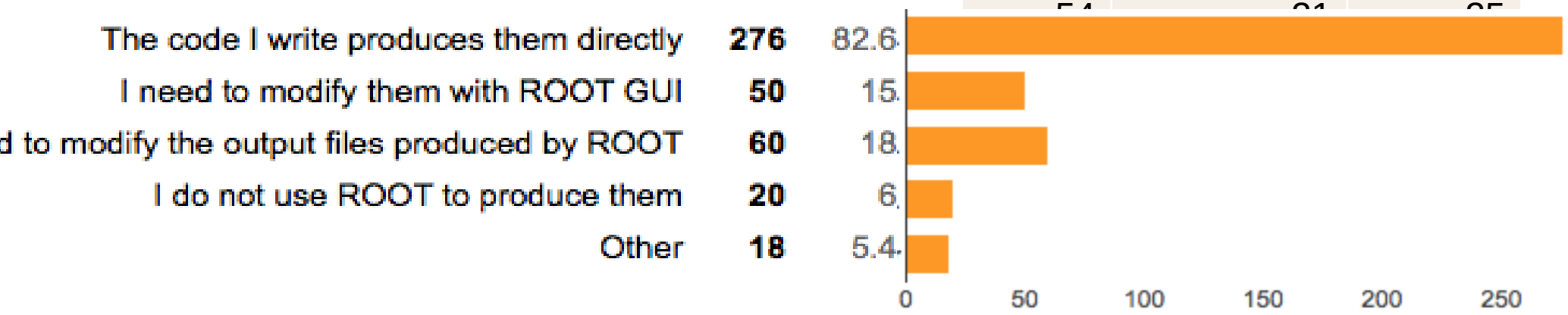
Graphics

	Don't know	Don't Use	Use
TStyle	3	6	91
Optimising the gStyle in my rootlogon.C	9	27	64
PS/EPS output	4	25	71
PDF output	2	7	91
SVG output	27	60	13
LaTex (TTexDump) output	36	44	20
PNG output	1	16	83
GIF output	5	68	27
Animated GIF output	36	52	12
JPEG output	6	68	25
Legend (class TLegend)	3	4	94
3D scatter plot, 3d surfaces	7	45	48

Graphics

	Don't know	Don't Use	Use
TH2Poly representation	75	20	5
TLatex	6	8	86
TMathText	36	20	43
TSpectrumPainter options	77	19	4
OpenGL specific drawing options	76	17	7

Producing publication ready plots



GUI, Event Display

	Don't know	Don't Use	Use
Editor (buttons on left of a canvas)	12	27	61
Guide Lines	58	27	15
QtRoot (from BNL)	81	16	3
QtGSI (from GSI)	87	13	<1
GUI Builder	71	26	3
Root Browser	2	8	90
Recorder	87	13	0
Root Geometry classes	63	26	12
Root Event Display TEve	63	29	8

I / O

	Don't know	Don't Use	Use
TKey	36	20	44
TMessage	77	19	3
TBufferFile	72	17	10
TStreamerInfo	60	26	14
I/O customisation rules	81	13	6
TVirtualCollectionProxy	88	11	1
TMemFile	88	9	3
TParallelMergingFile	89	10	1
TZIPFile	90	9	1
XML format	67	21	12
JSON format	75	17	7
HDF5 format	82	14	4
FITS format	84	12	3
XROOTD	39	22	40
HDFS	82	15	3

Trees Analysis

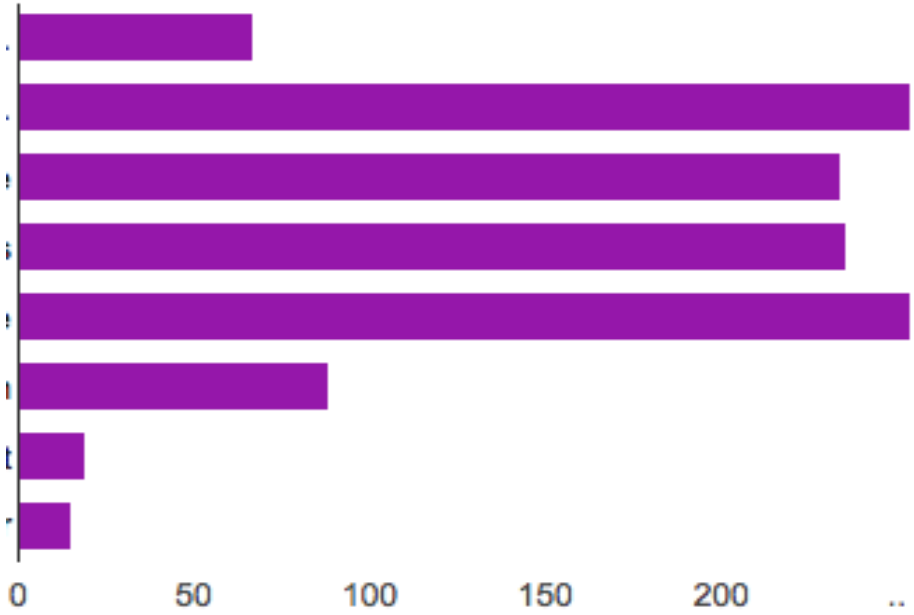
	Don't know	Don't Use	Use
TTree	1	1	98
TNtuple	17	31	52
TChain	2	7	91
TEntryList	38	27	35
TBundle	83	15	2
TTree::Show	29	16	56
TTree:Scan	12	12	76
TTree:Draw	6	6	89
TTree:Viewer	36	25	39
TCut	24	33	44
TGCut	60	34	6
TSelector	60	34	33
MakeCode	53	33	14
MakeClass	14	29	57
MakeSelector	37	33	30

Trees Analysis

	Don't know	Don't Use	Use
MakeProxy	78	18	4
TTreeReader	54	29	17
SetBranchAddress	9	13	78
TTreeCache	58	19	24

How did you start with ROOT?

ROOT's courses	**67**	19.7%
Reference Guide (class documentation)	**254**	74.7%
User's Guide	**234**	68.8%
Colleagues	**236**	69.4%
Existing code	**254**	74.7%
Forum	**88**	25.9%
Mailing list	**19**	5.6%
Other	**15**	4.4%



PROOF:selector

	don't know	don't use	use
How to create it	71	10	20
Use of TTree::MakeSelector	64	13	22
How to define input parameters	71	11	19
How to define outputs	71	10	19
Use of TProofOutputFile	79	9	12
Processing by object	86	8	6

PROOF: miscelanea

	don't know	don't use	use
use for generic tasks, not tree driven	80	13	7
use of feedback functionality	88	9	3
Add Merge() to a class	84	10	7
TProof::LogViewer	84	10	6

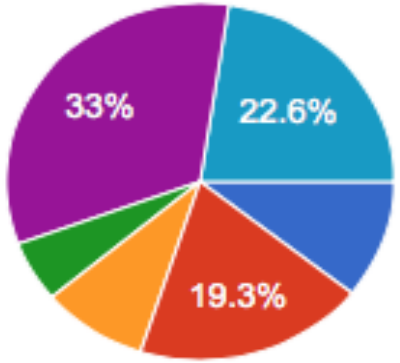
General comments about PROOF

- PROOF has a very steep learning curve and I would like to see improved tutorials focused on PROOF-lite.
- Used it, but difficult to maintain and debug
- Some failures produce no error message
- PROOF seems very, very buggy and unstable.
- Never managed to get it running in compiled code, without using CINT; stopped using it.
- Great, but PoD needs more support or updates, or its successor needs to come out soon.
- it would be nice if I could setup and start the jobs on my own, and only use PROOF to schedule which events get run on which node and to manage the I/O
- Most of the tasks I'm working on are trivial parallel, no need for PROOF

Usability of interfaces

	Lost	see when run	okay	mastering
TList	12	37	34	16
std::unique_ptr	35	34	15	16
pointers and those arrows	2	14	30	54
template classes	22	29	29	20
Variadic templates	56	28	10	6
standard algorithms e.g. std::for_each	15	27	35	23

PyRoot Performance / need for PyPy

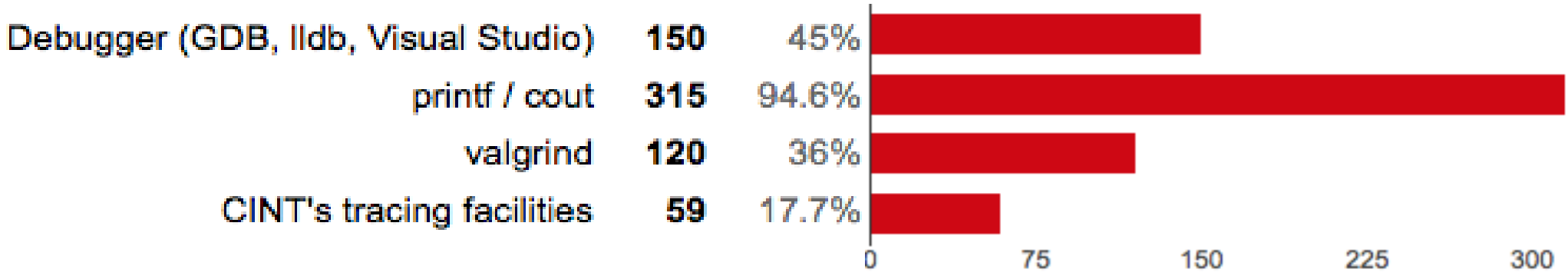


Is good enough	**35**	10.7%
Is fine, but I would welcome improvements but not at the expense of compatibility	**63**	19.3%
Is fine, but I want improvement through use of PyPy, accepting limitations	**29**	8.9%
Needs PyPy to get closer to C++ before it becomes useful	**18**	5.5%
Don't/won't use PyROOT	**108**	33%
What is PyPy?	**74**	22.6%

Editing Features

	Never noticed	don't care	useful
Syntax Highlighting	17	9	74
Detection of invalid code	16	11	74
Tab completion	5	7	88
Multi-line input	21	24	56

How do you debug?



Interactivity: what would increase your productivity most?

Better interactive prompt	**52**	16.7%
A "ROOT-IDE": an integrated environment that knows ROOT and your code	**71**	22.8%
Better recovery from errors	**128**	41%
Better printing of objects' values at the prompt	**39**	12.5%
Other	**22**	7.1%

Comments on things to improve - 2

- I think more fully embracing modern C++ styles is necessary to keep ROOT relevant in the future.
- Building standalone applications and not having to rely so much on CINT has become much easier over the years, and I would very much like to see this continue.
- I would like to see memory management improved, which I recognize is no small feat.
- Memory management - never obvious what who owns the object, especially in standalone app which deals with many ROOT files.
- Class hierarchy, especially for objects like histograms, is very frustrating
- I want to be able to use ROOT classes as easily as I would any STL class
- Removal of the need to write dictionary files (at least by hand), especially when working with python and C++ templates
- I think that instead of aiming to provide everything itself, ROOT should at most wrap external libraries to standardise their interface.
- Consistent naming of functionality. Why should some accessors start with Get and others not?
- ROOT unnecessarily wraps a lot of features that are part of C++ (e.g. maths libraries) when it would be better to leverage the existing features which are already industry-hardened
- Using strings instead of proper arguments (e.g. histogram drawing) has no place in modern C++ code

Comments on things to improve - 3

- I would like to see it broken apart into many smaller libraries, so that other codes could depend only on the tiny, tiny part of ROOT that they actually use -- and then work on removing that dependency.
- “Improve code design. Simple things like using template classes instead of having inheritence from TH1 for example. Using namespaces …and getting rid of silly aspects due to bad use of inheritance
  - it makes no sense for a TH1F to have a z axis for example. “
- Desire for backward compatibility has hampered the development of ROOT in some areas. Whilst this is essential for data storage in ROOT formats, for analysis code it could be argued that some changes or deprecation of old functions would be manageable changes that could improve the framework overall.
- “Compatibility" with PAW mentioned earlier is not necessary - maintaining the ability to convert a PAW file into a ROOT file is important, but other "compatibility" issues, where they can be decoupled from a purely file storage standpoint, should no longer be a design principle for ROOT development.
- I have not yet used ROOT 6 much, but I am finding the changes in the C interpreter interesting, I am in favour of the stricter interpretation of the code in the interpreter, and my initial impression is that this is a positive step.

Histograms

- Histogram ownership in TDirectory is confusing and error prone.
- TH2Poly class causes major headache (several inherited TH1 methods do nothing?).
- Would greatly appreciate more stl support (i.e. std::string instead of TString, std::vector instead of TList,...)
- The variable-bin size in histograms. The histograms should draw not the sum of weights in a bin, but the sum of weights divided by the width of the bin, so that when the bins have variable-size, the distribution that is seen is continuous, and has a physical meaning. Sometimes a user wants just labels, not values on the bin. So when the bin have sizes, an option to draw one of the other would be useful.

Statistical Analysis

- Statistical analysis can be a real can of worms, but 90% of all cases require either a simple least-squares or liklihood minimization.
- There should be easy entry-level ways that one can correctly handle baysian limit setting or basic hypothesis testing.
- All these basic tools should be in the main ROOT package with excellent documentation and the more sophisticated and obscure tests and methods should not dominate the beginners guide
- I think the machine learning field progress requires to start up a team to improve and further develop TMVA package.
- Concerning RooFit, it would be nice to have more examples to do p0 and exclusion plots.
- Gaus+Landau Convolution. There is no clear information about it, parameters -- what exactly every parameter mean

PyRoot

- Improve speed and more convenient handling of object ownership
- Better support for 2D arrays in pyroot.
- Easier way to write to TTree in pyroot.
- pypy documentation - more modular installation, see IPython -> Jupiter - better support of std::vector
- Had success running pyroot in an ipython notebook server using rootnotes module (see link); setup a notebook server running on my group's cluster, connect to server through SSH tunnel and a browser on my laptop. Very nice way to browse my data and create quick visualizations, and prototype the analysis code.
- Make the Python interface more pythonic, separate data storage and data visualisation (e.g. why does the TH1 contain and display the data???),

Plotting

- Why not use an existing plotting suite and minimally interface it to ROOT classes?
- Fiddly and unpredictable behavior in plotting: difficult to set text sizes in a consistent way between different TPads on the same canvas. Enormous amounts of text produced on any crash make it difficult to debug.
- Need better font support, cleaner interfaces to libraries like RooFit, TMVA.
- In creation of pub. quality plots have to mix visual settings with data extraction. It is just a mess.
- Ideal work pattern would be to make default drawing -> perfect it in interactive session -> and save visual settings for every future use. ……
- Customizing graphical output especially with multiple TPads
  - it is very difficult to scale e.g. fonts in different pads to be of the same size
- Contours in 2D plots
- Plotting with RooFit - Use of gif or svga. In general time evolving graphics …
- Plot style. It takes too much time to set style for plots
- More advanced TTree::Draw features (e.g. stack different cuts).
- More automatic plot features
  - e.g. automatic legend placement, automatic frame range when drawing multiple objects.

Plotting continued

- Plotting infrastructure must be improved. For example, text size should scale with the canvas not the pad
- Plotting of TGraph (it's crazy!): difficult to setup limits on axes, dots are plotted in random order and axis have random range
- Plotting of multiple histograms on the same plot: they get shifted in respect to each other when I set up range on the first histogram, and I have to zoom the axis by hand in prompt ROOT to get rid of the shift.
- Usage of gStyle, canvas and pads updating and modification and so on: the order of the commands is really important and you never know why. Sometimes have to do h1->Draw(), h2->(Draw("same")) and again h1->Draw("same")
- Transparency does not look to be correctly handled with pdf and eps - The legend should be placed automatically on the canvas such that it does not hide other points / graphs, titles. - The default filling color for graph should be empty.
- With PAW you can make perfectly-looking histograms right out of the box. Root needs alot of adjustments to make histograms look nice.

Plotting continued

- Simpler differential histograms. Funcitons like drawing a spectrum in units of per-keV is very common for me, but requires a loop over all bins and actually changing the bin contents.
- TGraphs treated as first-class citizens (e.g., graph.Draw() shouldn't do nothing)
- Better-behaved TChains. To loop through a TChain with a TEntryList requires different syntax than a bare TTree; TChains leave files open and can eventually trip over maximum files limit.
- Better thread-safe behavior. Heavy use of globals, especially in the GUI classes, makes multi-thread operation require lots of synchronization and protection, sometimes in unexpected places.
- Graphics editing for final plots: - automatically include templates from experiments - example plots with code to copy but of publication quality - a button to click to add a ratio plot (I know how to do it by hand, but it takes a while and this is a common application)
- Consistency: How do I change the Y axis on my plot? I know the answer, but it depends on the plot. Either I have to use GetYaxis()->SetRangeUser, SetMaximum and SetMinimum, or SetLimits. Multiply this kind of problem by the amount of classes that ROOT has and you will see the frustration we all suffer.

Plotting continued

- ROOT color palette functionality is confusing. I managed to hack in some of the nice palettes from the seaborn python module, but I confess that I still don't understand how the palettes work.

GUI

- Please use Qt !
- Improve usage from python (already very good, can be improved)
- Native latex support, including publication quality equation in plots.
- I would like more GUI functionality for doing common analysis tasks like making cuts on variables.

Documentation

- The manual needs to be user friendly. This can be achieved by finding out the top 100 (and then 1000) things we do and having a twiki page that goes through them. For instance: 1. Histogram n points 2. Graph n points. With and without errors. 3. Color the plots and make various snazzy plots we see in talks.
- Arrange items by topic from a physicist's point-of-view. For instance: 1. Reading data (from a file, from a tree, from ... is not relevant) 2. Fitting data (1-D, 2-D, 3-D, 4-D, ...) 3. Statistics 4. ... Currently we have to wade through pages of the manual to figure anything out.
- RooUnfold better support and documentation
- Explanations of what functions do, specifically it should be consistent across all functions in all classes.
  - “I like it very much when you have a 5 page explanation of a function but hate it when you have absolutely no explanation of an equally complicated function in the same class”
- Important information should not be 'hidden' in the documentation.
  - e.g. In TTree::Draw you mention halfway through the explanation that the result can be accessed using gPad->GetPrimitive("htemp"). This takes a LONG time to figure out if you don't know what you are doing. Perhaps you can make a section in the documentation for "common uses".

Tutorials

- The tutorials have a great deal of information that nicely showcase examples of how to do almost anything in ROOT. Unfortunately, they are not organized in a user friendly manner, I think it may be alphabetical?
- A better interface to the examples that are already available would be extremely helpful.
- Being able to know what is contained in each tutorial without randomly clicking until one finds an example would be excellent.

Miscellaneous

- Greater promotion of PROOF would be useful. Can I use it to advantage on my 8-core machine? Do I need to completely re-write everything or re-learn how to use ROOT?
- There is a program called RStudio for R. It really is excellent. I would love to see such a thing for ROOT. mathematica-like notebook functionality (or at least knitr like document compiler) would also be quite welcome.
- The CMake based build system is still harder to use than the simple ./configure method. The possible options that the CMake build can take are hard to find, and in general, it's very hard to make CMake disregard libraries that it finds on the build machine. (Making it hard to generate minimal/generic libraries.)
- Make reading/writing branches automatic so that I can code only the logic. More meaningful errors instead of segfaults. You can make use of C++11 features like lambdas or at least dictionaries to read/write from the tree. Event should be loaded in the memory as a safe object rather than plain C arrays and variables.

What do you like most about Root

- Very large set of statistical tools.
- TTree and RooFit
- Generates beautiful plots very quickly
- It's very easy to get up and running. Making plots is straightforward.
- It has everything I've ever needed.
- IO functionality - TFile and TTree Histograms and drawing PyROOT fully compatible with ROOT and python TMinuit
- It's easy to get some results, plotting data is simple.

Like less: ‘a user’s comments on usability’

- “I use ROOT as much as I have to and no more. It can take dozens or sometimes even hundreds of lines of code to produce a reasonable plot (e.g. with stacked histograms and a data/MC ratio plot with error shading). This is an absurd situation for an established library which is used largely for this purpose. I would like to see interfaces modernised according to modern coding practices rather than conventions inherited through fortran or years of "just getting things done". In particular, common tasks like plotting need to be easy otherwise there is little point in using it.”

Formal Training – this is new

- The ROOT team are preparing 3 ROOT courses for inclusion in the CERN Training Programme
  - see : https://root.cern.ch/root-training-proposal
- Basic Course
  - the interpreter, histograms, files, trees, fitting, python interface, GUI
- Advanced Analysis Course
  - RooFit, RooStats, multi-variate analysis, PROOF
- Advanced Developers Course
  - rootcore, geometry, event display, httpserver, javascript (JSROOT), ROOT as a service (ROOTaaS)
- Input is welcome on finalizing course structure and topics