Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bundesministerium
für Bildung
und Forschung

ROOT tutorial

— part II —

1st April 2015

Adrian Perieanu

ROOT tutorial: goals

* how to install it ✓

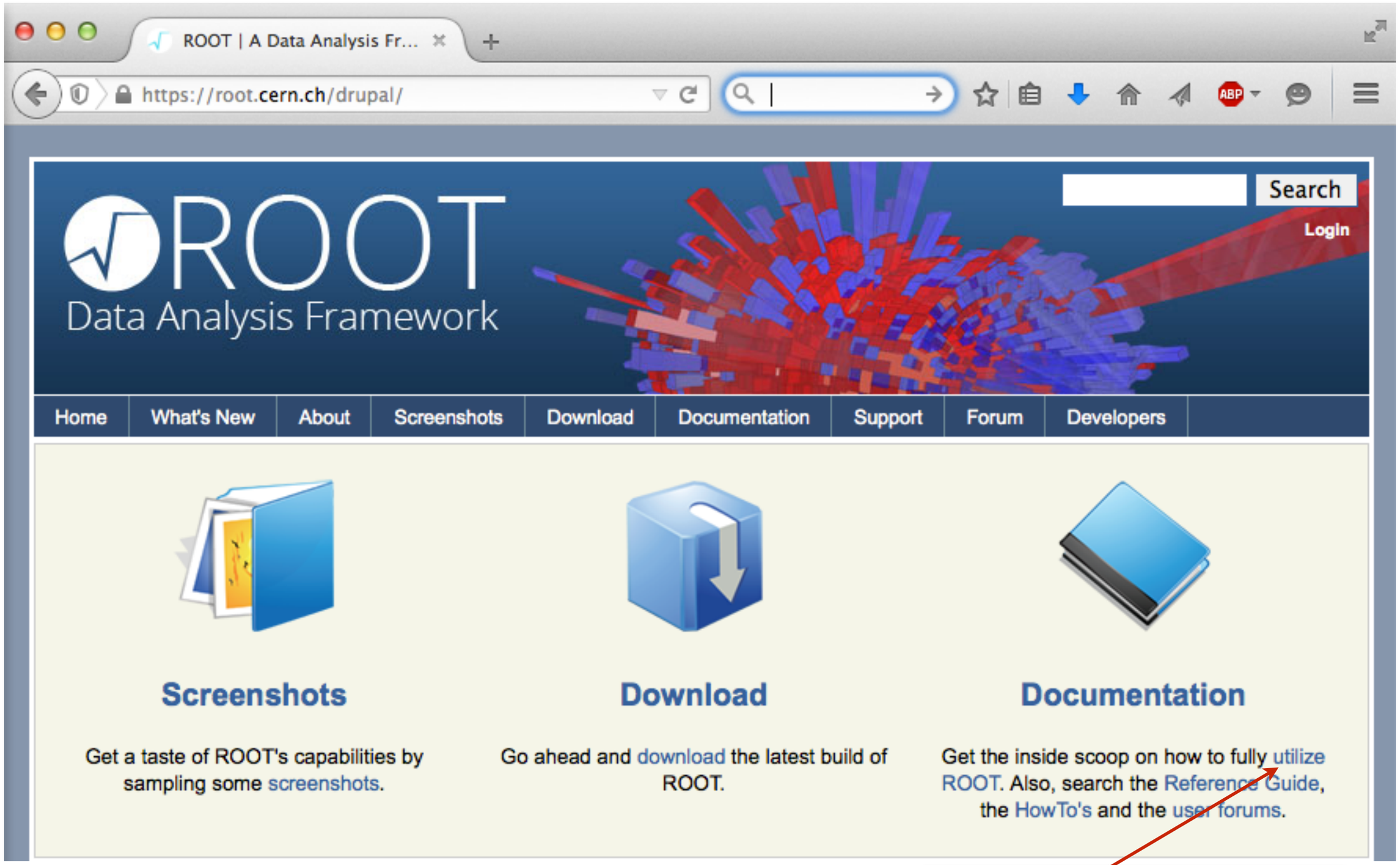
* how to find/read documentation ✘

* perform an interactive analysis with ROOT ✘

* design and write own analysis macros ✘

* how to store results of your analysis ✘

ROOT tutorial:
how to find/read documentation



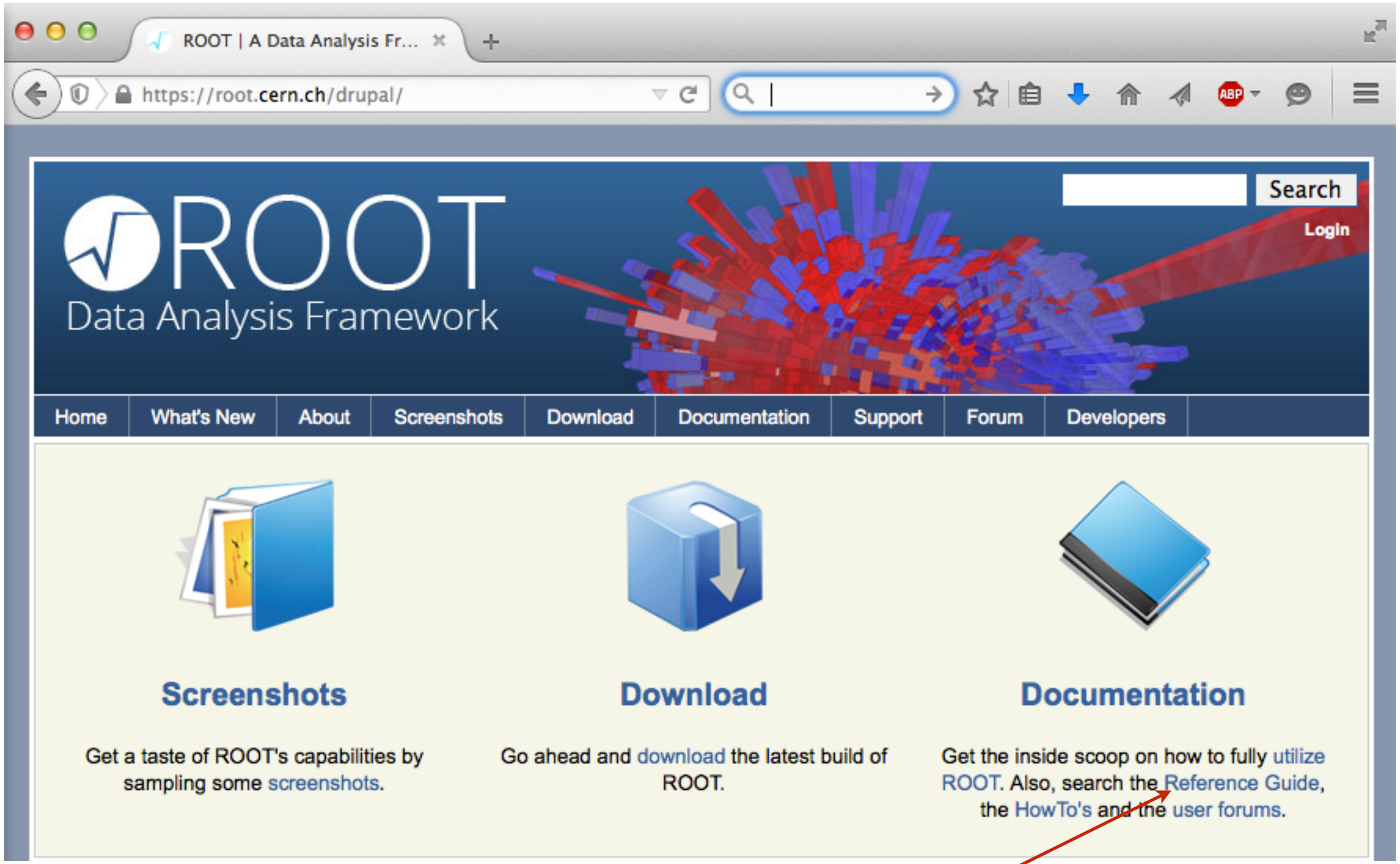
we start from the same point as before

ROOT documentation: I. read

a series of manuals are waiting for you

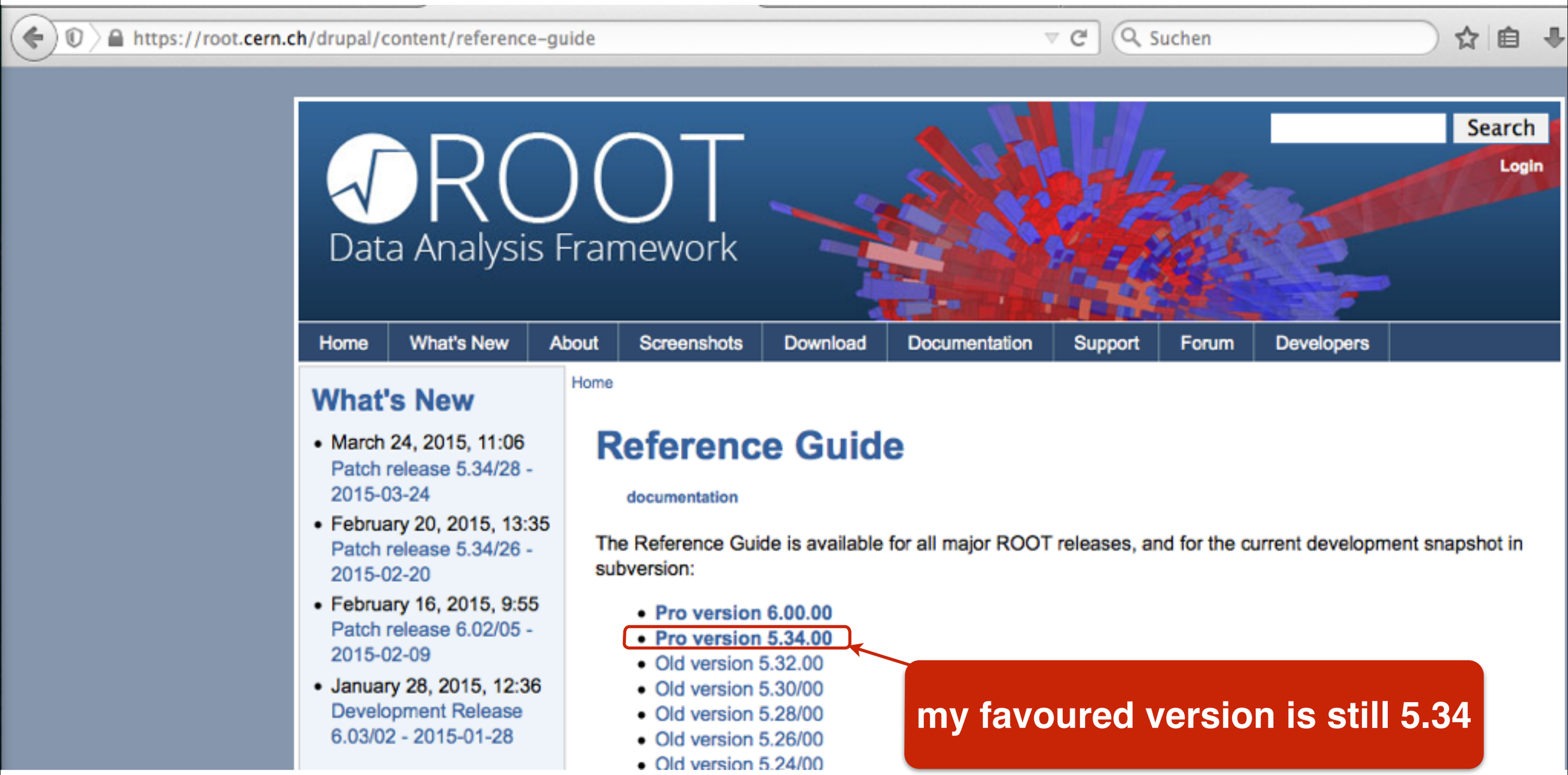
if you prefer not to read them, go to next page

ROOT tutorial: I'. don't read

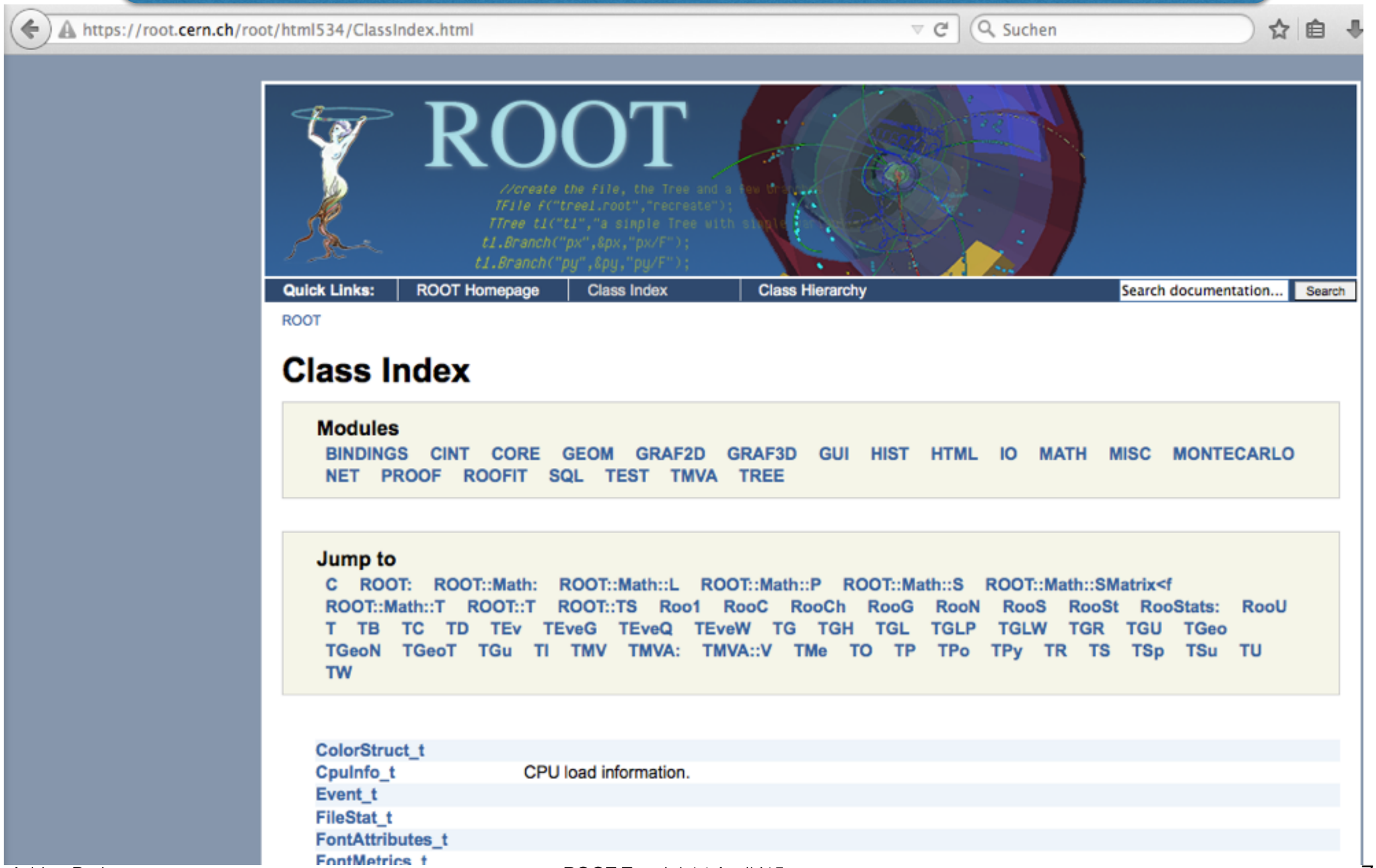


go to "Reference Guide"

ROOT documentation:
II. choose your favourite version



ROOT installation: II. choose



ROOT installation: III. search

https://root.cern.ch/root/html534/ClassIndex.html

Suchen

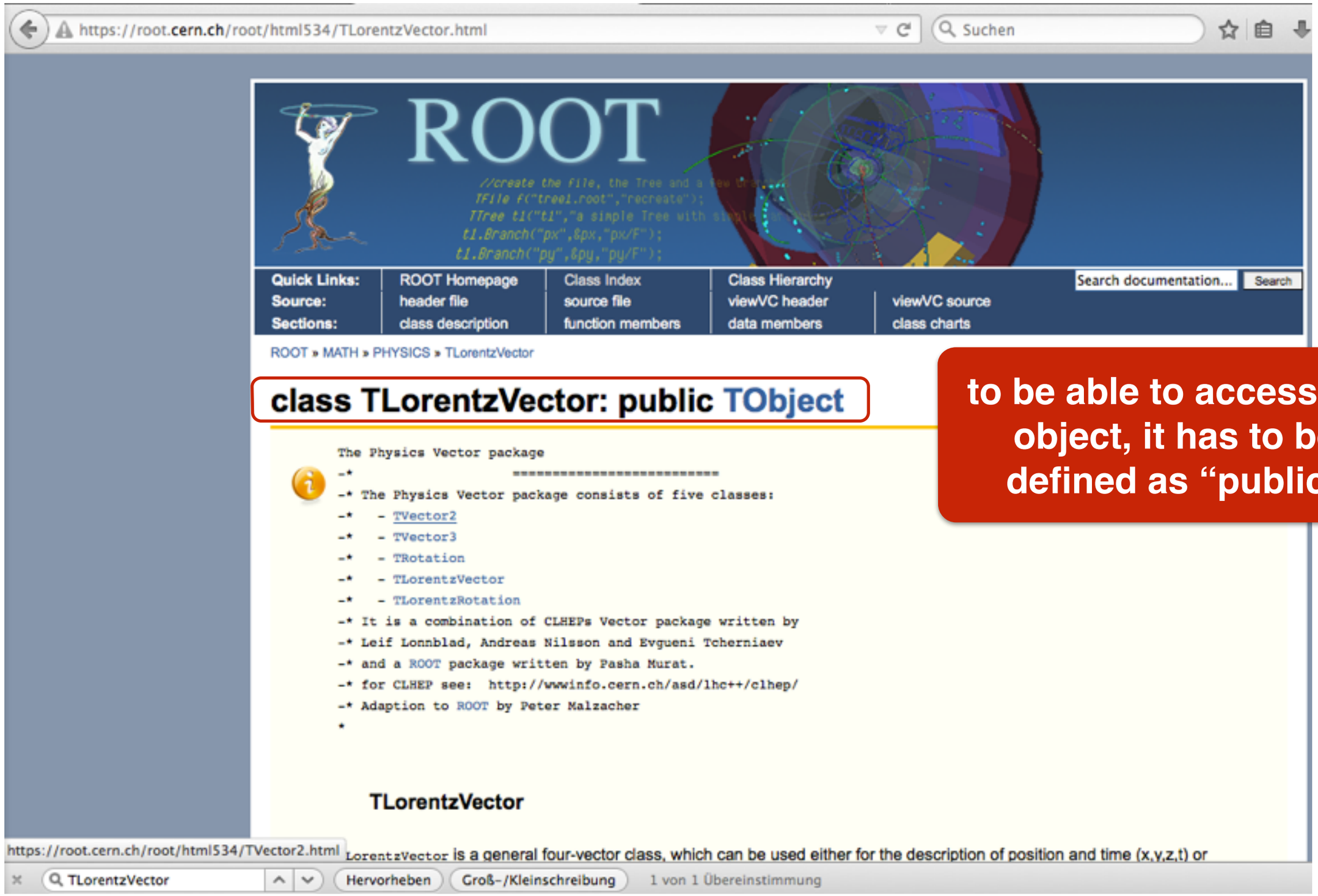
TLeafF	A TLeaf for a 32 bit floating point data type.
TLeafI	A TLeaf for an Integer data type.
TLeafL	A TLeaf for a 64 bit Integer data type.
TLeafO	A TLeaf for an 8 bit Integer data type.
TLeafObject	A TLeaf for a general object derived from TObject.
TLeafS	A TLeaf for a 16 bit Integer data type.
TLegend	Legend of markers/lines/boxes to represent obj's
TLegendEntry	Storage class for one entry of a TLegend
TLibraryDocInfo	documentation for a library
TLimit	Class to compute 95% CL limits
TLimitDataSource	input for TLimit routines
TLine	A line segment
TLineEditor	GUI for editing Line attributes
TLinearFitter	fit a set of data points with a linear combination of functions
TLinearGradient	Linear gradient fill.
TLinearMinimizer	Implementation of the Minimizer interface using TLinearFitter
TLink	Link: hypertext link to an object
TList	Doubly linked list
TListIter	Linked list iterator
TLockFile	Lock an object using a file
TLockGuard	Exception safe locking/unlocking of mutex
TLorentzRotation	Lorentz transformations including boosts and rotations
TLorentzVector	A four vector with (-,-,-,+) metric
TMCVerbose	Verbose class for MC application
TMD5	MD5 cryptographic hash functions with a 128 bit output
TMLPAnalyzer	A simple analysis class for MLP
TMVA	
TMVA::BDTEventWrapper	
TMVA::BinarySearchTree	Binary search tree including volume search method
TMVA::BinarySearchTreeNode	Node for the BinarySearchTree
TMVA::BinaryTree	Base class for BinarySearch and Decision Trees
TMVA::CCPruner	
TMVA::CCTreeWrapper	
TMVA::Config	Singleton class for global configuration settings
TMVA::Config::IONames	
TMVA::Config::VariablePlotting	
TMVA::Configurable	Virtual base class for all TMVA method
TMVA::CostComplexityPruneTool	
TMVA::CrossEntropy	Implementation of the CrossEntropy as separation criterion

ctrl+f

https://root.cern.ch/root/html534/TLorentzVector.html

TLorentzVector | Hervorheben | Groß-/Kleinschreibung | 1 von 1 Übereinstimmung

ROOT installation: IV. inspect



to be able to access the object, it has to be defined as “public”

ROOT installation: IV. inspect some more

https://root.cern.ch/root/html534/TLorentzVector.html

Suchen

TLorentzVector

`TLorentzVector` is a general four-vector class, which can be used either for the description of position and time (x,y,z,t) or momentum and energy (px,py,pz,E).

Declaration

`TLorentzVector` has been implemented as a set a `TVector3` and a `Double_t` variable. By default all components are initialized by zero.

```
TLorentzVector v1;      // initialized by (0., 0., 0., 0.)
TLorentzVector v2(1., 1., 1., 1.);
TLorentzVector v3(v1);
TLorentzVector v4(TVector3(1., 2., 3.),4.);
```

For backward compatibility there are two constructors from an `Double_t` and `Float_t` C array.

Access to the components

There are two sets of access functions to the components of a `LorentzVector`: `X()`, `Y()`, `Z()`, `T()` and `Px()`, `Py()`, `Pz()` and `E()`. Both sets return the same values but the first set is more relevant for use where `TLorentzVector` describes a combination of position and time and the second set is more relevant where `TLorentzVector` describes momentum and energy:

```
Double_t xx =v.X();
...
Double_t tt = v.T();

Double_t px = v.Px();
...
Double_t ee = v.E();
```

The components of TLorentzVector can also accessed by index:

```
xx = v(0);       or     xx = v[0];
yy = v(1);              yy = v[1];
zz = v(2);              zz = v[2];
tt = v(3);              tt = v[3];
```

scroll-down the page and you will see a series of examples

ROOT installation: IV. inspect some more

https://root.cern.ch/root/html534/TLorentzVector.html

Suchen

Function Members (Methods)

public:

```
                TLorentzVector ()
                TLorentzVector (const Double_t* carray)
                TLorentzVector (const Float_t* carray)
                TLorentzVector (const TLorentzVector& lorentzvector)
                TLorentzVector (const TVector3& vector3, Double_t t)
                TLorentzVector (Double_t x, Double_t y, Double_t z, Double_t t)
       virtual ~TLorentzVector ()
      Double_t Angle (const TVector3& v) const
      Double_t Beta () const
          void Boost (const TVector3& b)
          void Boost (Double_t, Double_t, Double_t)
       TVector3 BoostVector () const
static TClass* Class ()
      Double_t CosTheta () const
      Double_t DeltaPhi (const TLorentzVector& v) const
      Double_t DeltaR (const TLorentzVector& v) const
      Double_t Dot (const TLorentzVector& q) const
      Double_t DrEtaPhi (const TLorentzVector& v) const
      Double_t E () const
      Double_t Energy () const
      Double_t Et () const
      Double_t Et (const TVector3& v) const
      Double_t Et2 () const
      Double_t Et2 (const TVector3& v) const
      Double_t Eta () const
       TVector2 EtaPhiVector ()
      Double_t Gamma () const
          void GetXYZT (Double_t* carray) const
          void GetXYZT (Float_t* carray) const
virtual TClass* IsA () const
      Double_t M () const
```

scroll-down even more, and you will find the public methods

ROOT installation: let's try to use it

define 2 Lorentz Vectors

add the 2 Lorentz Vectors

extract the mass

```
perieanus-MacBook-Pro% root -l
root [0] TLorentzVector blume(1.,1.,1.,1.);
root [1] TLorentzVector baum(1.,1.,1.,2.);
root [2] TLorentzVector baum_hat_blume;
root [3] baum_hat_blume = baum+blume;
root [4] Double_t masse_blume = blume.M();
root [5] Double_t masse_baum_hat_blume = baum_hat_blume.M();
root [6] cout<<"masse_blume:"<<masse_blume<<" masse_baum_hat_blume:"<<masse_baum_hat_blume<<endl;
masse_blume:-1.41421 masse_baum_hat_blume:-1.73205
root [7] .q
```

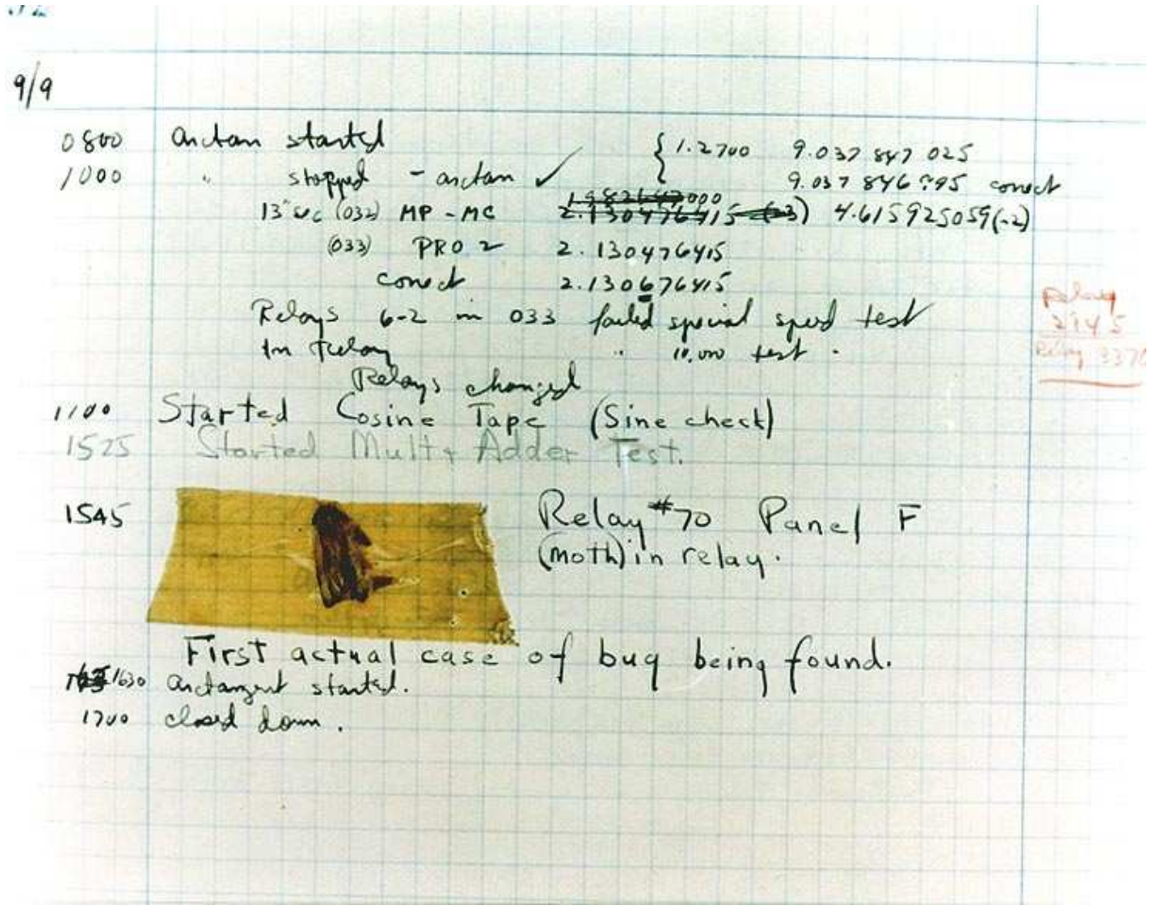
print out the mass

is the result correct?
how can you check it?
why is it important to check it?

every line of written code introduces potentially a (software) bug

what is a (software) “bug”?

what is a (software) “bug”?



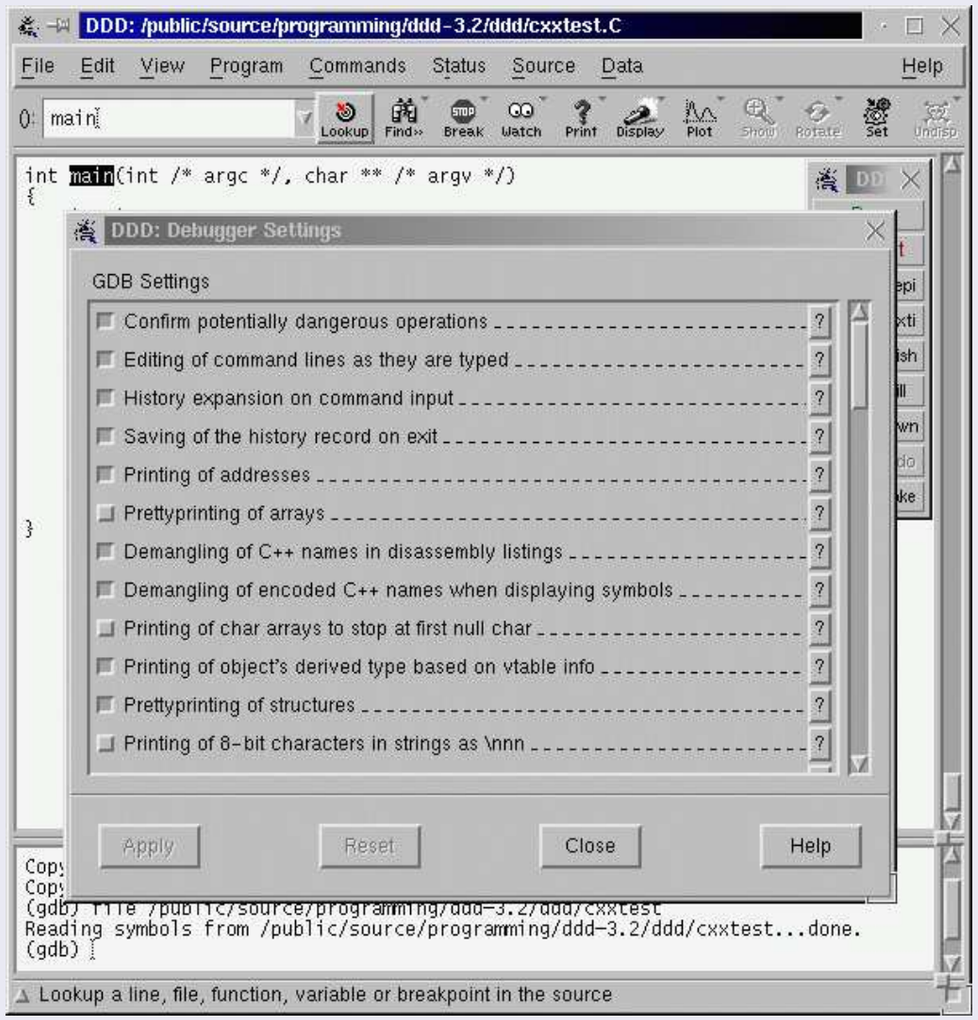
The first bug was really a bug: a moth which in 1947 went into a "big" computer at the Harvard University (more than 15 m long, 72 Byte memory, three additions per second) and made a shortcut

Someone took the bug and "posted" in the logbook (a very special way of Debugging...)

there are also debugging tools

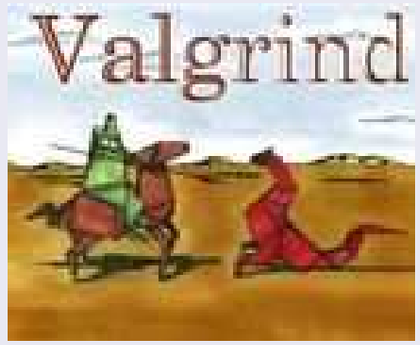
gdb

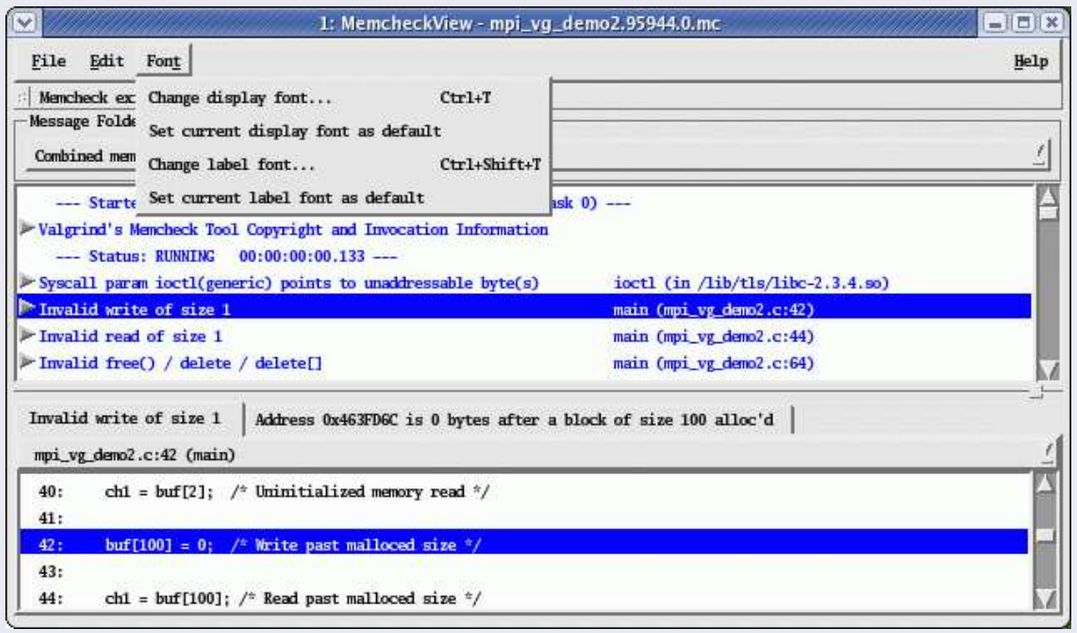
- gdb - der GNU-Debugger, a Unix tool, with its user interface, ddd



Valgrind

- valgrind for x86-Linux Programs





the best way is when one understands its own bugs!

bugs will become your day by day companions

some of the famous bugs:

- 26.09.1983: A Soviet Satellite indicated that 5 ballistic missiles were launched from US.
- Possible Consequence: the World War III.
- But: the Soviet duty officer reasoned that if the U.S. was really attacking, more than five missiles would have been lunched.
- A bug in the Soviet software failed to filter alarms caused by sunlight reflecting off cloud-tops.

- Sometime the bugs have also "good" sides

- 2001 at the Open-Air-Festival in Sziget (Ungarn): an ATM gave double the amount of the requested money, but subtracted from the account the requested amount (as written on receipt...)

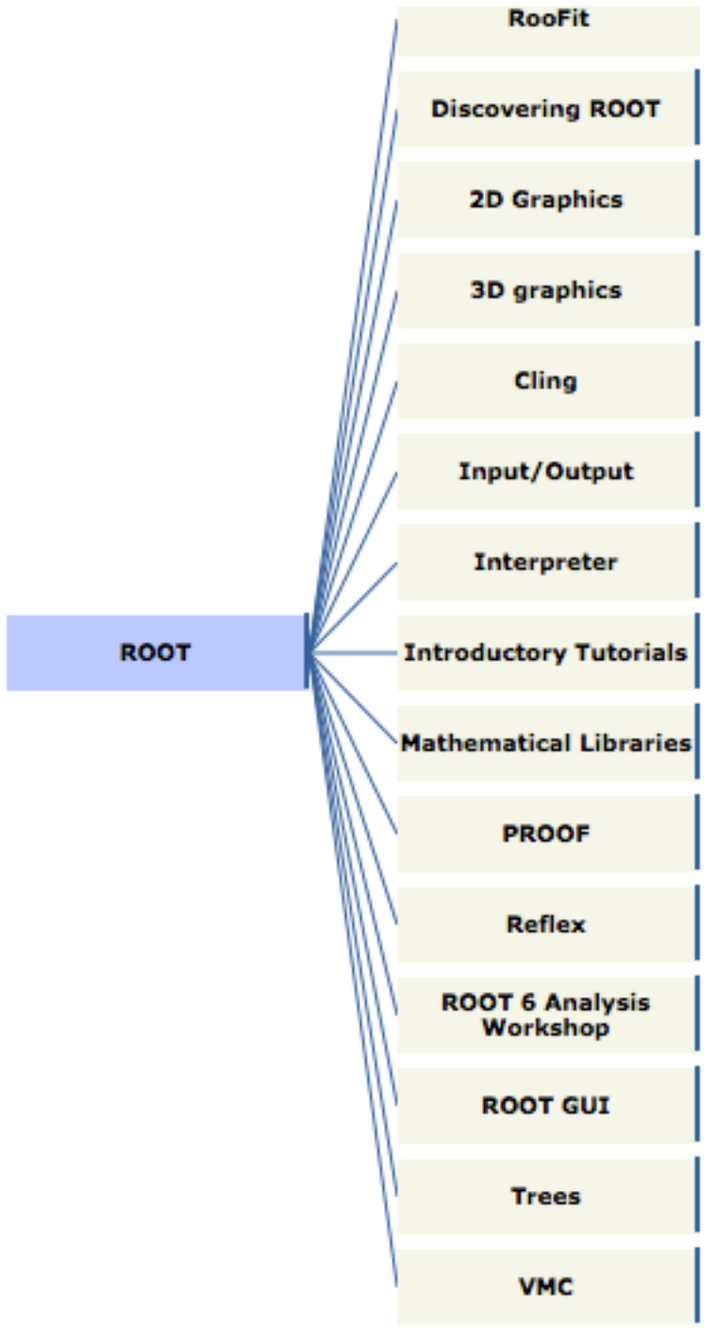
summary: documentation

* go to: https://root.cern.ch/drupal/content/documentation

* if inpatient:
https://root.cern.ch/drupal/content/reference-guide

* choose your favoured version and search for the class you need

* once you have implemented your code, check it for bugs
(you better find them before you start writing your thesis 😄 😄 😄)



time for a break and (a lot of) questions

almost half way through, from now on we can play with data