Using R in ROOT with the ROOT-R package

Lorenzo Moneta (CERN),
Omar A. Zapata M. (GSOC student)

What is R ?

- R is an open source language and environment for statistical computing and graphics.
  - open source implementation of S language developed by J. Chambers
    - R popularity and usage increased largely in recent years
- R provides a large variety of statistical tools
  - linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …
- Environment is highly extensible with a large number of existing packages

Introduction

- ROOT-R package
  - a new ROOT package to interface with R environment
    - to use R, its functions and tools from the ROOT prompt or any C++ application
    - give access to ROOT users to the R capabilities and its rich functionality
  - Developed by O. A. Zapata (GSOC student 2014)

C++ Interface to R

- ROOT-R make uses 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 extra layer to Rcpp and RInside
  - new classes to facilitate conversion between ROOT/C++ objects and R objects
  - easy to call R functions from ROOT prompt and C++ code

Goal of ROOT-R

- Give easy access to ROOT users to R tools
  - more focused on using the statistical tools rather than graphics tools from R
- To further integrate R in ROOT plug-ins based on R can be developed
  - ROOT::Math::Minimizer plug-in using R optimisation packages (RMinimizer)
  - Plan to develop TMVA methods based on R classification packages (GSOC project for 2015)

How does it work ?

- TRInterface: main class interfacing to R
  - static instance that can be made available at the ROOT prompt
  - can be used to execute R commands

```
root[0] auto r = ROOT::R::TRInterface::Instance();

root[1] r.Parse("print(version$version.string)");
[1] "R version 3.1.0 (2014-04-10)"
```

R Object -> ROOT

- TRObjectProxy class for converting from R objects to ROOT/C++ objects

```
root[] ROOT::R::TRObjectProxy rop = r.ParseEval("c(1,2,3,4)");
root[] std::vector<double> v = rop.As<std::vector<double>>();
```

- Better to use operator= for convenience

```
root[] std::vector<double> v = r.ParseEval("c(1,2,3,4)");
root[] TMatrixD m = r.ParceEval("matrix(c(1,2,3,4),2,2)");
root[] std::array<double,10> a = r.ParceEval("seq(1:10)");
```

- Or TRInterface::operator[] using the name of the R object

```
root[] r.Parse("mat=matrix(c(1,2,3,4),2,2)");
root[] TMatrixD m = r["mat"];
```

Using Operator << and >>

- We can also use operator<< to pass an R command to the TRInterface

```
root[] r << "mat<-matrix(c(1,2,3,4),2,2)";
```

- And use operator >> to transform an R object into a ROOT/C++ object

```
root[] TMatrixD m;
root[] r["mat"] >> m
```

ROOT/C++ -> R

- ROOT/C++ objects can be passed to R using the operator[] of TRInterface

```
root[] std::vector<double> v = {1,2,3,4,5};
root[] r["v"] = v;
root[] r.Parse("print(v)");
[1] 1 2 3 4 5
```

- or in combination with operator<<

```
root[]TMatrixD A(2,2,v.data() );
root[] r["A"] <<  A;
root[] r << "print(A)");
     [,1] [,2]
[1,]    1    2
[2,]    3    4
```

Supported Object Conversions

- C++ fondamental types (int, float,double)
  - R scalar (vectors of size 1)
- std::vector, std::array, std::list, TVectorD
  - R vector types
- TMatrixD ↔ R matrix
- Direct conversion from/to more complex R object (e.g. list or data frames) is not supported yet

Passing Functions to R

- If you have free C functions

```
Double_t myfun(Double_t x){
    return 2*cos(x);
}
Double_t myfun2(const std::vector<Double_t> & x){
    return x[1]*cos(x[0]);
}
```

- you can use the class TRFunction to pass it to R

```
r["dilog"]<<ROOT::R::TRFunction(TMath::DiLog);
r["myfun"]<<ROOT::R::TRFunction(myfun);
r["myfun2"]<<ROOT::R::TRFunction(myfun2);
r << "print(dilog(0))";
r << "print(myfun(0))";
r << "print(myfun2(c(0,4)))";
```

- Can be used for passing functions to numerical algorithms from R (Numerical integration, minimisation, etc…)

Example: Integration in R

- Integrate a function in R.
  Note R uses vectors as inputs to functions

```
std::vector<Double_t>  BreitWignerVectorized(const std::vector<Double_t> & xx){
  std::vector<Double_t> result(xx.size());
  for(Int_t i=0;i<xx.size();i++)
  {
    result[i]=TMath::BreitWigner(xx[i]);
  }
  return result;
}

void Integrate() {

  auto r = ROOT::R::TRInterface::Instance();
  r["BreitWigner"]=ROOT::R::TRFunction(BreitWignerVectorized);
  Double_t value=r.ParseEval("integrate(BreitWigner, lower = -2, upper = 2)$value");

  std::cout<<"Integral of the BreitWigner in [-2, 2] =" << value << std::endl;
}
```

Passing C++ Classes to R

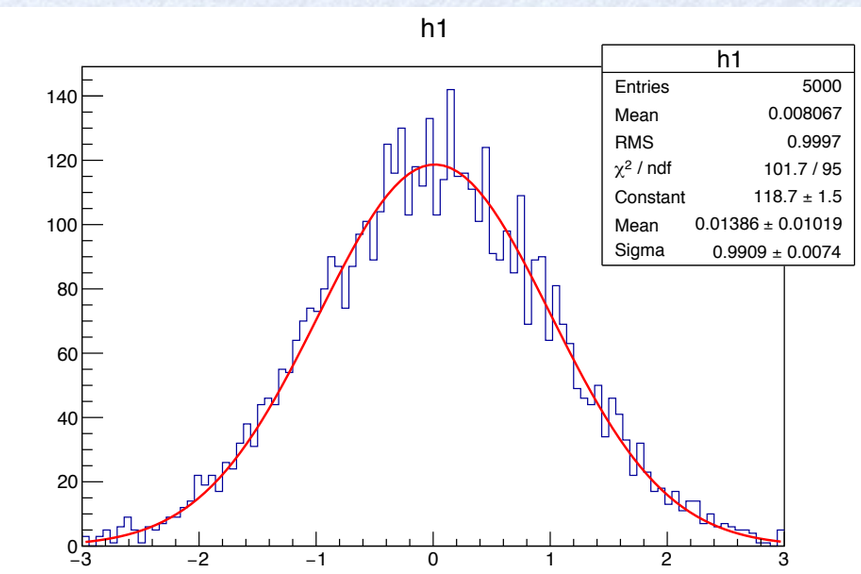
- It is also possible to pass a C++ class from ROOT to R
  - Requires defining a R module advertising the class methods in the R environment
  - ROOT-R facilitate this by providing some helper cpp macros
- More information available at ROOT-R online documentation

RMinimizer

- ROOT plugin for Minimisation implemented using R
  - give access to R optimisation tools when fitting or multi-dimensional function minimisation
    - based on R optim/optimx packages
    - Class developed by Kirby Hermann (GSOC student 2014)

```
ROOT::Math::MinimizerOptions::SetDefaultMinimizer("RMinimizer","L-BFGS-B");
hist->Fit("gaus");
```

```
root [4] h1.Fit("gaus")
Value at minimum =101.673
****************************************
Minimizer is RMinimizer / L-BFGS-B
Chi2                      =      101.673
NDf                       =           95
NCalls                    =          265
Constant                  =      118.694   +/-   1.47659
Mean                      =    0.0138555   +/-   0.0101907
Sigma                     =     0.990906   +/-   0.00741443
```



Future Developments

- Add wrapper classes (plugin) in ROOT for R classification tools
  - integrate some of R classification tools in TMVA
    - GSOC project for 2015
- Extend ROOTR to map also R data frames

Conclusions

- ROOTR provides easy access to R tools in ROOT and C++
- Easy to use directly R from ROOT prompt
  - extend the functionality of ROOT
    - can be used also to verify and cross-check some of existing ROOT math and stat tools
    - new ROOT plugins or wrapper classes implemented using R can be easily created
- Package is ready do be released in next ROOT production version (6.0.4)

Documentation

- ROOT-R User Guide
  - http://gfif.udea.edu.co/web/tiki-print.php?page=ROOTR%20Cling%20Users%20Guide
- ROOT-R Twiki page
  - http://gfif.udea.edu.co/web/tiki-index.php?page=ROOTR+Cling