Restructuring of the numpy integration in PyROOT

Fantastic features and where to find them

Enric Tejedor and Stefan Wunsch

ROOT

Data Analysis Framework

https://root.cern

Laziness with cppyy

- Cppyy allows to register pythonizations that are executed **lazily**

```
import ROOT
t = ROOT.TTree()
```

triggers

```
def pythonizor_ttree(klass, name):
  if name == 'TTree':
    # pythonize klass!
```

- But this only works for **classes**
  - Not for class templates
  - Not for namespaces

```
import ROOT
ROOT.VecOps
```

no way to lazily inject
behaviour (e.g. new functions)

Slimming down “import ROOT”

- We currently inject a few PyROOT features in C++ namespaces
- Problem: we don’t do it lazily!
  - The injections happen at “import ROOT” time
  - They trigger the loading of many extra libraries (e.g. libTMVA)
- Particular cases discussed on next slide

Current status: injected features

- ROOT supports full interoperability with numpy for RVec, RTensor and RDataFrame
- The features are attached to the respective namespaces and lack visibility
- The current integration via the namespace make it impossible to include them fully lazily

```
# Numpy to RVec
npy = numpy.array(...)
rvec = ROOT.VecOps.AsRVec(npy)

# RVec to numpy
npy = numpy.asarray(rvec)

# Numpy to RTensor
npy = numpy.zeros(shape)
rten = ROOT.TMVA.Experimental.AsRTensor(npy)

# RTensor to numpy
npy = numpy.asarray(rten)

# Numpy to RDataFrame
data = {"x": numpy.array(...), "y": ...}
df = ROOT.RDF.MakeNumpyDataFrame(data)

# RDataFrame to numpy
data = df.AsNumpy()
```

Proposal: reorganization

- Move all numpy interoperability features to the `ROOT.Numpy` “namespace”
- Allows **lazy loading**, cuts down libraries loaded during `import ROOT` by half
- One place to find them all, simple grouping by purpose, better to document
- PR WIP: https://github.com/root-project/root/pull/5252

```
# Numpy to RVec
npy = numpy.array(...)
rvec = ROOT.Numpy.AsRVec(npy)

# Numpy to RTensor
npy = numpy.zeros(shape)
rten = ROOT.Numpy.Experimental.AsRTensor(npy)

# Numpy to RDataFrame
data = {"x", numpy.array(...), "y": ...}
df = ROOT.Numpy.MakeDataFrame(data)
```