RooFit Development - Intuitive Python bindings for RooFit

Project Mentors

- Jonas Rembser
- Lorenzo Moneta

Student Developer

- Harshal Shende, B.Tech (EEE)
  National Institute of Technology Karnataka, India

This presentation outlines the planned work on the pythonization of the RooFit library.

Transition from C++ to Python for RooFit, with aims to simplify complex workflows and enhancement of the python interface, greatly reducing the amount of code that has to be written.

Making RooFit easier to use in Python addresses a clear need of the user community.

Project Objectives

To Pythonize all RooFit functions that accept RooCommandArgs, such that they take keyword arguments instead using the Python standard way to pass optional arguments to functions, and enhance the Python interface to make it more “pythonic”, i.e. easier to use.

For example, currently :

```
pdf.fitTo(data, ROOT.RooFit.Range("sideband"))
```

Pythonic :

```
pdf.fitTo(data, range="sideband")
```

Improve the existing interface for major RooFit classes like :

- RooRealVar
- RooAbsReal
- RooDataHist
- RooDataSet
- RooRealIntegral
- RooArgSet
-

Another Example :

```
gauss.plotOn(frame, ROOT.RooFit.LineColor(ROOT.kRed))
```

Simplified

```
gauss.plotOn(frame, LineColor=”r”)
```

Sample code illustrating Pythonization

```
import cppyy
from ROOT import pythonization

def __getter(k, v):
    # helper function to get CmdArg attribute from `RooFit`
    # Parameters:
    # k: key of the kwarg
    # v: value of the kwarg
    if isinstance(v, (tuple, list)):
        attr = getattr(cppyy.gbl.RooFit, k)(*v)
    elif isinstance(v, (dict, )):
        attr = getattr(cppyy.gbl.RooFit, k)(**v)
    else:
        attr = getattr(cppyy.gbl.RooFit, k)(v)
    return attr


def _classFunc(self, *args, **kwargs):
    # Redefinition of `RooClass` for keyword arguments.
    # the keywords must correspond to the CmdArg of the function.
    # Parameters:
    # self: instance of class
    # *args: arguments passed to `classFunc`
    # **kwargs: keyword arguments passed to `classFunc`
    if not kwargs:
        return self._OriginalFunc(*args)
    else:
        nargs = args + tuple((__getter(k, v) for k, v in kwargs.items()))
        return self._OriginalFunc(*nargs)
```

Project Objectives

Simplification of the Complex Workflows and adding advanced pythtonizations in PyRoot. Make RooFit in pyROOT less verbose and more pythonic. For example,

- In the case where a std::unordered_map is expected, a pythonization that accepts a Python dictionary
- Functions that take a RooArgList or a RooArgSet can be Pythonized to accept a simple Python list.

```
dsmap = ROOT.std.map('string, RooDataSet*')()

dsmap.keepalive = list()

# from python dictionary to std::unordered_map
for c, d in dsdict.items():
    dsmap.keepalive.append(d)
    dsmap[c] = d

ds = ROOT.RooDataSet("data","data", ROOT.RooArgSet(x), Index=cat, Import=dsmap)
```

Aim to pass python dictionaries directly to `Import`

```
ds = ROOT.RooDataSet("data","data", ROOT.RooArgSet(x), Index=cat, Import=dsdict)
```

Project Objectives

Documentation and Adding Tests for the pythonizations and relevant changes to RooFit in tutorial directory (pythonizing recurring non-pythonic patterns).

Documentation at function level for the pythonizations, not only as class level using doxygen or docstrings.

Relevant Links

- https://root-forum.cern.ch/t/combining-roodatasets-in-pyroot/43615
- https://cppyy.readthedocs.io/en/latest/pythonizations.html
- https://github.com/root-project/root/pull/7753
- https://github.com/root-project/root/issues/7217