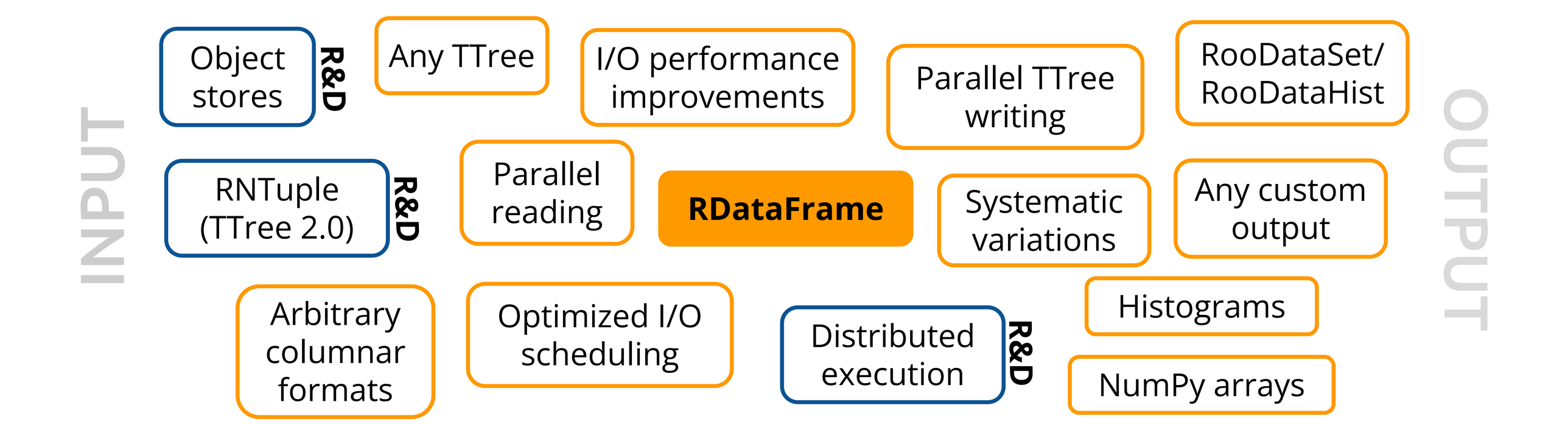
RDataFrame enhancements for HEP analyses

E. Guiraud, J. Blomer, S. Hageboeck, A. Naumann, V. Padulano,
E. Tejedor, S. Wunsch (EP-SFT, CERN)



1. RDataFrame in a nutshell

RDataFrame enables simple and fast HEP analysis in C++ and Python. The same programming model works for a single histogram or thousands, from a laptop to batch execution on a cluster (also see this talk).

The following lines produce a skimmed dataset and a control plot in a multi-thread event loop:

v6.14

```
ROOT.EnableImplicitMT()
df = ROOT.RDataFrame(tree, files)
df = df.Filter("pts[abs(etae) < 1].size() > 0")
       .Define("myVec", myFunc, ["pts"])
h = df.Histo1D("myVec")
df.Snapshot("newtree", "newfile.root")
```

2. Definition of per-sample values

DefinePerSample evaluates a quantity that depends on the sample being processed. Useful e.g. to define different event weights for MC and data:

v6.26

```
df.DefinePerSample("weight",
    [](unsigned slot, const RDF::RSampleInfo &s) {
        return s.Contains("MC") ? 0.5 : 1.; })
  .Histo1D("value","weight");
```
C++

3. Column redefinition

The recently introduced **Redefine** method makes it possible to modify the value and/or type of a column before further processing:

v6.26

```
df.Redefine("x", [](double  x) { return float(x)*0.5; }, {"x"})
  .Snapshot("tree", "newfile.root")
```
C++

4. Systematic variations

Starting from ROOT v6.26, RDataFrame provides a natural and flexible syntax to define systematic variations that automatically propagate to selections, derived quantities and results.

Here we add up/down variations of pt and fill a histogram with a quantity that depends on pt. We automatically obtain three histograms in output:

v6.26

```
nominal_hx =
   df.Vary("pt", "RVecD{pt*0.9, pt*1.1}",  ["down", "up"])
     .Filter("pt > k")
     .Define("x", someFunc, ["pt"])
     .Histo1D("x")

hx = ROOT.RDF.VariationsFor(nominal_hx)
hx["nominal"].Draw()
hx["pt:down"].Draw()
```
Python

5. Injecting Python into the C++ event loop through Numba

Through Numba's just-in-time compilation of Python functions, it is possible to inject Python logic into RDataFrame's fast C++, multi-thread event loop:

v6.24

```
@ROOT.Numba.Declare(["RVecD","RVecD"], "RVecD")
def good_pts(pts, etas):
   return pts[np.abs(etas) < 1]

df.Define("good_pts", "Numba::good_pts(pts, etas)")
```

```
# the code above will soon just be:
df.Redefine("pts", lambda pts, etas: pts[np.abs(etas) < 1])
```

Also at ACAT 2021

"Distributed RDataFrame: leveraging Dask and latest optimisations"
"RNTuple Performance: Status and Outlook"