Analysis tools
The selling points of our analysis tools

**Programmability** Be the simplest way to write down an analysis of data in ROOT files
- Example: Users doing exploratory analysis
- “Show me the $p_T$ distribution of the leading jet in the new NanoAOD production, now!”

**Performance** Be the fastest way to process ROOT files
- Example: Users running in production on large datasets
- “Produce histograms of the final observable for all datasets taken in 2016, 2017 and 2018 including all systematic shifts.”

What does it tell to our users about ROOT if we fail in these two points on our own data format?
Know our users and their applications

Provide excellent support of common data structures used at the LHC experiment
- Example: NanoAOD-like end-analysis ntuples
- Example: Jagged arrays (see next slide)

Provide concise solutions to common use-cases
- Example: Provide a two-liner to draw two histograms and a ratio plot below
- Example: Four-vector arithmetic following an array programming paradigm

Who are our most important users?
- For whom do we want to optimize our workflows?
- Importance of LHC experiments vs individual users?
Nasty nested loops problem

**Problem** Treatment of multi-dimensional (jagged) arrays without indexing
- Jagged arrays: Arrays with variable length (like Muon_pt[nMuon])
- Multi-dimensional arrays: Arrays with variable number of dimensions (like a numpy array, but probably jagged!)

**Question** How to manipulate such objects without explicit loops and index magic?
- Problem that injects a lot of complexity in user’s code
- Awkward array already gives a good idea of such a programming model
- Optimal programming model still under vivid discussion
- **Can we do better?**
Find a successor for PROOF How to support users running on large datasets using small clusters?
- Offer a system scaling from a laptop to tens of nodes
- Important field to investigate besides local parallelism

PyRDF Current proposal for distributed analysis with ROOT
- Modern technology (programming model, backends, ...)
- Still in R&D state, not feature complete
- Interesting topics to be followed up:
  Caching, additional backends, automatic merging of results
- Far-future goal: Full C++ support
Actively use analysis examples

**Analysis examples are crucial for proving the real-world applicability of our developments**

- Documentation: Explaining the tools to the user
- Feature completeness: Be sure that we cover common workflows
- Workflows: Find out by ourselves that a workflow is not optimal, not using our users as testers in first place
- Macro benchmarking: Know our performance for common tasks in high detail

**Collaboration with CERN Open Data portal could be the key**

- Already providing several open CMS analyses
- Early next year: 12 ATLAS analysis operating on 150GB of data to be released
Ease of installation
Best ways to install ROOT

We should not advocate ...
- ... installation of binaries
- ... compilation from source

The beauty of uproot: pip install uproot
- This is one of the major selling points for uproot

Make conda more visible
- Probably promoting to the default way?
- Provides fast installation independent of the platform

Having our own repositories for distro packages
- Supporting most popular package managers (yum, dfn, apt)
- Provides fast installation for a large user-base
Documentation about how to install ROOT is not good

- Not directly visible on the website
- Conda is missing
- **We need very clear instructions for the very beginners**

Usage of the source script is not obvious

- Due to rise of virtual environments such as used with Python?
- People are used to global installations?
  - →Install and work out-of-the-box is the expectation
- Could we improve the visibility and documentation?
Provide up-to-date docker images
- We have no consistent tags for releases
- Can also be a way of distributing ROOT

Docker could gain additionally in popularity if preservation becomes obligatory for analyses
- ROOT docker images could be perfect baseline
- ReANA developers already building ROOT docker images by themselves

Perfect baseline for testing
- We should test ROOT based on the Docker images
- Scales easily and covers wide range of users/distros
Actively promote SWAN as a way to distribute ROOT

- Highly accessible installation for anyone with a CERN account

People of the LHC experiments use SWAN actively

- Provide additional support for this important user base
Build system
ROOT is highly monolithic
- Heavy dependency for any project
- Cost of integration is high
- Supports spread of dependencies all over ROOT
- Example unsupported use-cases: Histograms without graphics

Modularity supports popularity
- Simplifies life of package maintainers
  → Splitting ROOT in sub-packages by the maintainer is highly non-trivial
  → See effort of conda maintainers
- Better packages → More seamless installation procedure
- Enables us to use public services more efficiently (CI/Travis, codecov, coverty, ...)
- Could be supported by an internal package manager
Building externals out of the build tree

- Shared between multiple builds
- Improves the turn-around cycle of our developments
- Goes in hand with better modularity

Quality of life improvement by external LLVM as default
We have our own implementation for everything
- Regex
- GUI, 3D
- Math
- XML
- ...

Moving these to a legacy library and replacing them by industry standards?
- Slims down ROOT
- Most likely improves test coverage
PyROOT
Main goal for PyROOT 2020: PyROOT experimental to become the **default**

- Target: 6.22
- Should take most of the effort

**Two main concerns:**

- Stability: test, test, test
  - ROOT, PyCool/PyCoral, LHCb, ...
  - [https://sft.its.cern.ch/jira/browse/ROOT-10347](https://sft.its.cern.ch/jira/browse/ROOT-10347)
- Backwards compatibility / feature completeness
  - [https://sft.its.cern.ch/jira/browse/ROOT-10418](https://sft.its.cern.ch/jira/browse/ROOT-10418)
PyROOT needs proper **documentation**
- Pythonizations, API, interactive graphics
- As much as we can before 6.22, the rest for 6.24

PyROOT should be **modular**
- Multiple Python packages, not just one
- Experimental PyROOT already addresses this, thanks to Massimiliano
  - ROOT pythonizations, cppyy packages (3), TPython, JupyROOT, JsMVA
  - [https://github.com/root-project/root/pull/4647](https://github.com/root-project/root/pull/4647)
Interoperability with Python data science ecosystem is a must
- Ongoing effort: AsMatrix, AsNumpy (+ pandas), AsRVec
- New proposals: definition of ufuncs

We need more pythonic interfaces
- Old pythonizations already been ported
- What area has more priority / lacks these interfaces more?
  - RooFit? I/O? RDataFrame?
PyROOT: the best way to make Python analysis efficient in any circumstance

We need to advertise better the power of PyROOT: not just any Python bindings!
- Dynamic jitting
- Usage of efficient C++ from Python
- Injecting “kernels” into RDF

Support **cling debug symbols**
- PyROOT would benefit from it (e.g. RDataFrame from Python using jitted code)
Windows
Fix remaining issues on Windows
Have ROOT working on Windows 64 bit
Have PyROOT experimental working on Windows
What is the priority/effort we want to give to Windows-related developments?