From hcana to Python, a possible roadmap (episode I)

Gabriel Niculescu
James Madison University

- **Motivation**
- **Scope**
- **(potential) Pitfalls**
- **Current Status**
- **Outlook**

Motivation

- ***… as per Leonardo di Pisa:***

1, 1, 2, 3, 5, 8,…

- ***The US produces enough PhDs to "renew" all R1 faculty every 4 years or so…***
- ***Most of the world does not do ROOT!*** ***(Or C++)***

- ***Python has a lot more traction in the corporate (and not only) world! (more job prospects…)***
- ***Lots of tools, active & dedicated community.***

- ***Why not? Scientific answer should be programming language independent!***

Scope (I)

- *Won't bother you with a work flowchart of a (typical) analysis, though that might have merit too!*
- *This is what I agreed/intend to do:*
- *Identify and implement a/some solution(s) for moving the final stage of the physics analysis to Python.*
- *NOTE1: Initial data reduction & calibration still in ROOT/C++*
- *NOTE2: Python analysis can work alongside or instead of the C++-based analysis.*
- *In case you were wondering:*
  - *Many high energy, particle physics experiments now have Python components (if they are not done 100% in Python!)*
  - *Since 2015… NSF has funded (Data-Intensive Analysis… - DIANA/HEP)*

Scope (II)

- ***To better clarify things I sat down and put thoughts on a piece of .txt file…***

- *% Plan for moving the JLab Hall C final analysis*
- *% from a ROOT/C++ framework to a Python framework*

- ***I. Assumptions:***
- *~~~~~~~~~~~~~~~~~*
- ***I.0. - User knows what they are doing/trying to do!***
- *I.1. - Raw data was reconstructed (using the best calibrations available)*
- *and the results (histograms, trees, etc.) were saved into ROOT files and*
- *(optionally) in text files (various report files for scalers, efficiency, etc.)*
- *I.2. - Only the final steps of the analysis (binning, acceptance and other corr.) are left.*
- ***II. Needs (for I.2):***
- *~~~~~~~~~~~~~~~~~~~~~*
- *II.1. - Traverse trees.*
- *II.2. - Apply cuts a/o weights.*
- *II.3. - Produce 1,2, 3(?)D histograms.*
- *II.4. - Plot histograms, graphs, etc.*
- *II.5. - Perform fits on histograms, graphs...*
- *II.6. - Extract numerical information (fit parmeters, tables, etc.)*

- **** want to do all of II.xx in Python! ****
- *III. Action items (to do List):*

Scope (III)

- *% Plan for moving the JLab Hall C final analysis*
- *% from a ROOT/C++ framework to a Python framework*

- *III. Action items (to do List):*
- *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*

- *Based on I. and II. above one needs to find a way to...*

- *III.1. - Read ROOT file in Python.*
- *III.2. - Convert ROOT hists, trees, etc. into suitable Python "structure(s)" (TBD).*
- *III.3. - Ensure permanence of Python "structure(s)" - i.e. SAVE them in a*
- *non-ROOT file.*
- *III.4.-.. - Do all the steps listed under II. above!*

- **** do all of the above reasonably fast! (x-check wrt ROOT!) ****

- ***Now, looking at the Python ecosystem (w/ an eye for Data Analysis tools) there is an obvious choice!***

Pitfalls (I)

- **Possible solutions for addressing JE data "issue":**

  1. *Carefully "prune" the ROOT tree, keeping only the needed branches/leafs.*
     - *This should result in a flat, 2D structure that will map real well in a DF.*
     - *(after all one only needs 7 values to fully define one particle, throw in a little PID info and FP quantities and we are still at only 2-3 dozen vars… for both spectrometers combined!)*
     - *This is the fastest solution.*

  2. *Keep the data "as is" and cope with the time/space/$$ penalty.*
     - *Requires the least amount of work now*
     - *Really poor outlook compared w/ 1 & 3.*

  3. *"Flatten" the ROOT tree by splitting it into several 2D structures:*
     - *Main DF w/ all single valued quantities of interest + pointers/indexes (and sizes) for all array-per-event variables*
     - *One DF for each of the array-per-event variables (use index and range to access)*
     - *Can keep all variables. Analysis code more involved – some speed penalty.*

Pitfalls (II)

- *Pandas DataFrame is really designed to work (very well!) with 2D data structures.*
- *Hall A/C ROOT trees have “jagged edge” (JE) structure*
- *i.e. 3D data, w/ the 3rd dimension variable.*

- *This poses a HUGE problem for Python/Pandas!*
- *… especially if one wants to ensure permanence.*
- *HDF5 (h5) file format (which is how one might want to save the data) has real difficulties handling JE data.*
- *It can be done but it is not pretty!*
- *Done some testing and the data ballooned by a factor of ~10 when trying to save it in h5. – Clearly needs more work!*

Status

- ***Pandas DataFrame is really designed to work (very well!) with 2D data structures.***
- ***Hall A/C ROOT trees have "jagged edge" (JE) structure***
- ***i.e. 3D data, w/ the 3rd dimension variable.***

- ***This poses a HUGE problem for Python/Pandas!***
- ***… especially if one wants to ensure permanence.***
- ***HDF5 (h5) file format (which is how one might want to save the data) has real difficulties handling JE data.***
- ***It can be done but it is not pretty!***
- ***Done some testing and the data ballooned by a factor of ~10 when trying to save it in h5. – Clearly needs more work!***

Status (II)

- ***Right now I can…***
- ***Read ROOT tree into numpy arrays.***
- ***Read ROOT tree into pandas DF (even w/ JE)***
- ***Once in the DF one can slice/plot/etc (matplotlib)***
- ***Save DF into .h5 file (huge penalty if JE data)***

- ***See also output file…***

Outlook

- ***Continue to work on the JE problem identified earlier***
- ***Come back w/ more quantitative assessment of the possible solutions.***
- ***Produce "publication quality" plots for the F2 exp…***

- ***(more distant future): pySIMC?***