JÜLICH
FORSCHUNGSZENTRUM

panda

Analysis with PandaRoot

IKP-1 PANDA Analysis Meet-Up

20/27 May 2015, Andreas Herten

Outline

- Important resources
- Simulation & Analysis flow
- Scripts'n'Tools
- Prometheus (SSH, Job Resubmission)

- My example channel:

  $\bar{p}p \rightarrow D^+ D^- \rightarrow K^-\pi^+\pi^+ \; K^+\pi^-\pi^-$

Resources & Prerequisites — General

- PandaRoot **SVN** Repository:

  https://subversion.gsi.de/trac/fairroot/browser/pandaroot

  – Create your directory in the **development branch**!

    My: https://subversion.gsi.de/trac/fairroot/browser/pandaroot/development/aherten

- Computing **Wiki**:

  https://panda-wiki.gsi.de/cgi-bin/view/Computing

- Computing **Forum**:

  https://forum.gsi.de/index.php?t=index&cat=26

- **Documentation**:

  http://cbmroot.gsi.de//panda_doc/daily/html/classes.html

Resources & Prerequisites — Analysis

- **Analysis How-To (Wiki)**

  https://panda-wiki.gsi.de/foswiki/bin/view/Computing/PandaRootRhoTutorial

- Analysis Presentation (PDF)

  https://panda-wiki.gsi.de/foswiki/pub/Computing/PandaRoot/Goetzen_PandaRootAnalysisNewRho_v4_exp.pdf

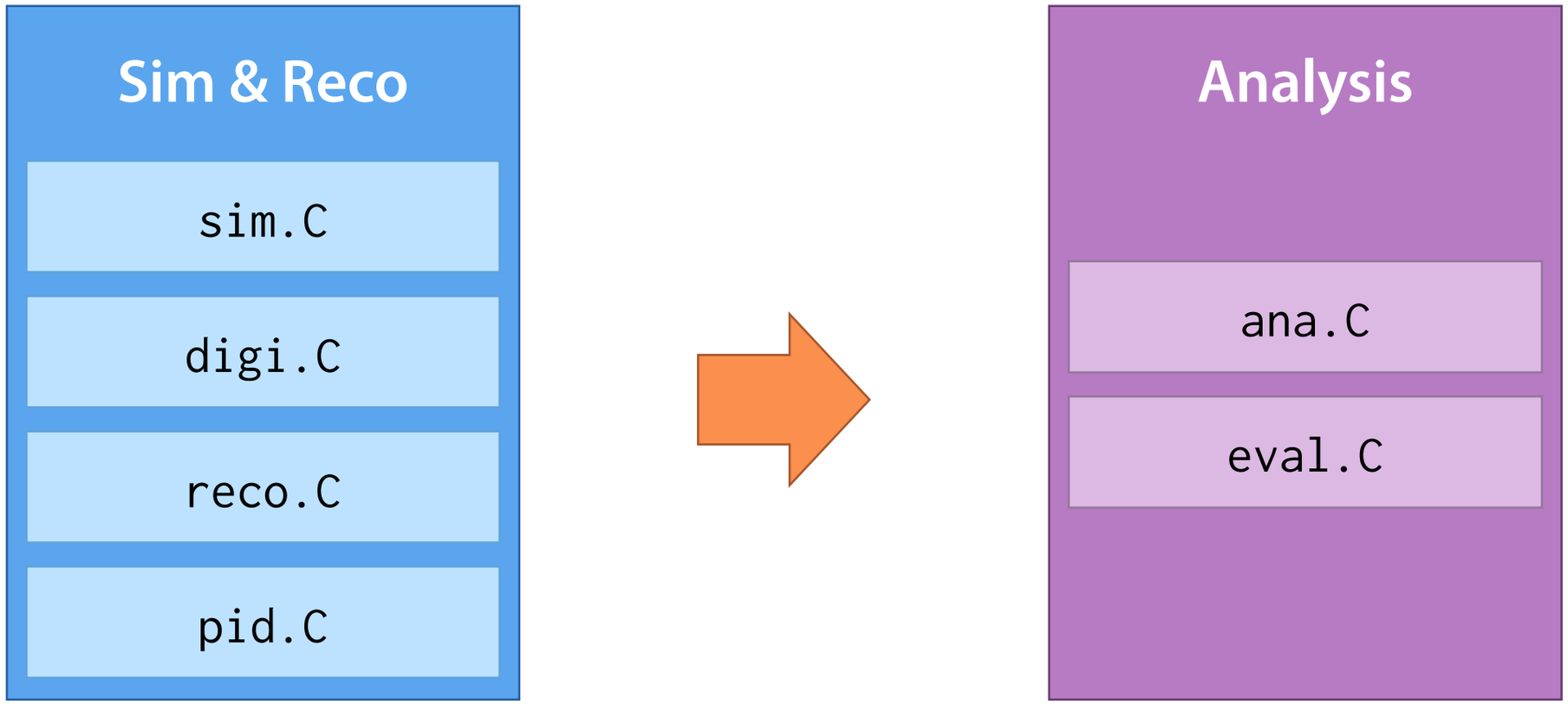
- Macros in `tutorials/rho/`

  https://subversion.gsi.de/trac/fairroot/browser/pandaroot/trunk/tutorials/rho

- `RhoCandidate` class documentation
  - Automatic: http://cbmroot.gsi.de//panda_doc/daily/html/classRhoCandidate.html
  - Detailed highlights: https://panda-wiki.gsi.de/foswiki/bin/view/Computing/PandaRootAnalysisJuly13
- Analysis Forum

  https://forum.gsi.de/index.php?t=thread&frm_id=152&

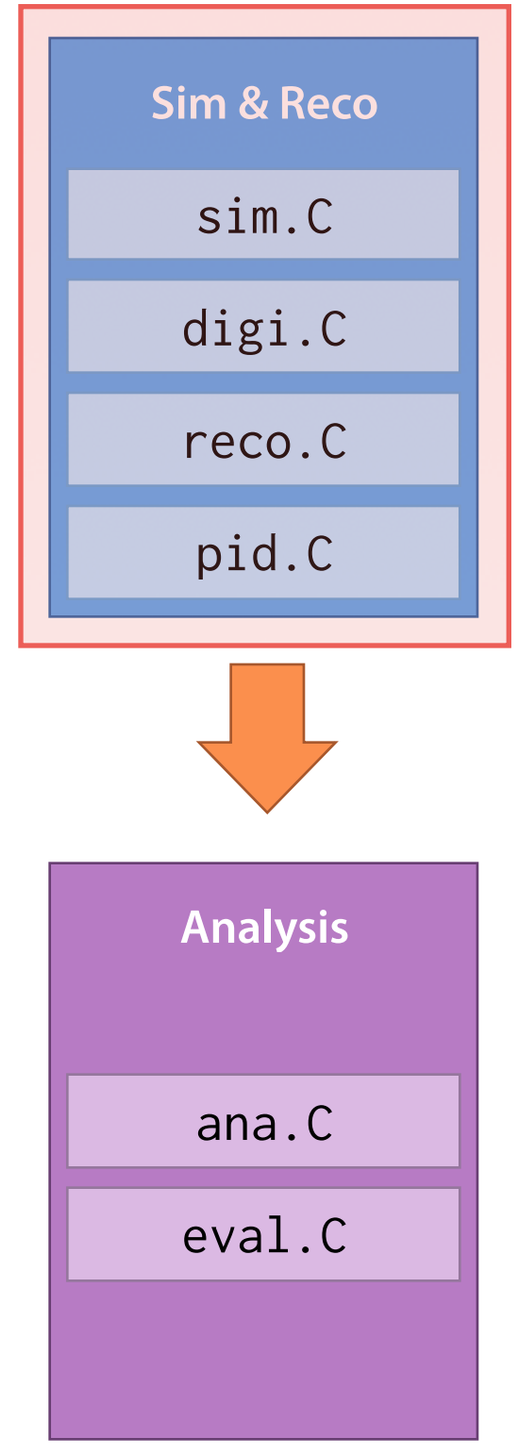
Simulation Flow



Sim & Reco

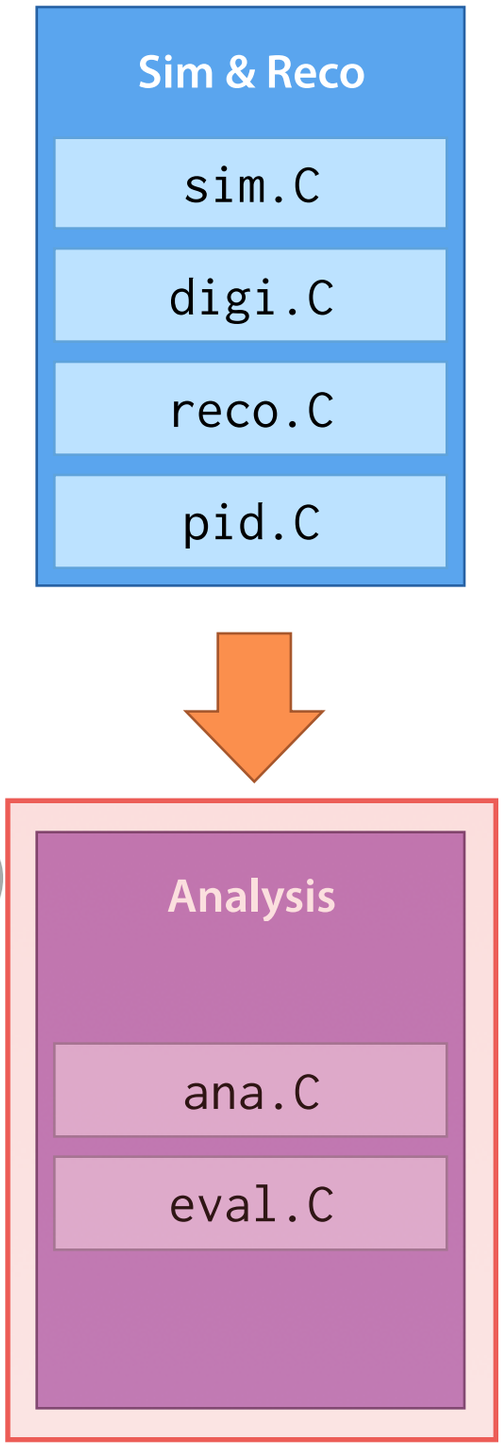
- Out of scope here, just one thing:
- Preparing for *Prometheus:* Add a prefix!

```
void sim(Int_t nEvents = 10, TString prefix = "") {
  TString OutputFile = prefix + "sim_complete.root";
  TString ParOutputfile = prefix + "simparams.root";
  // ...
}
```



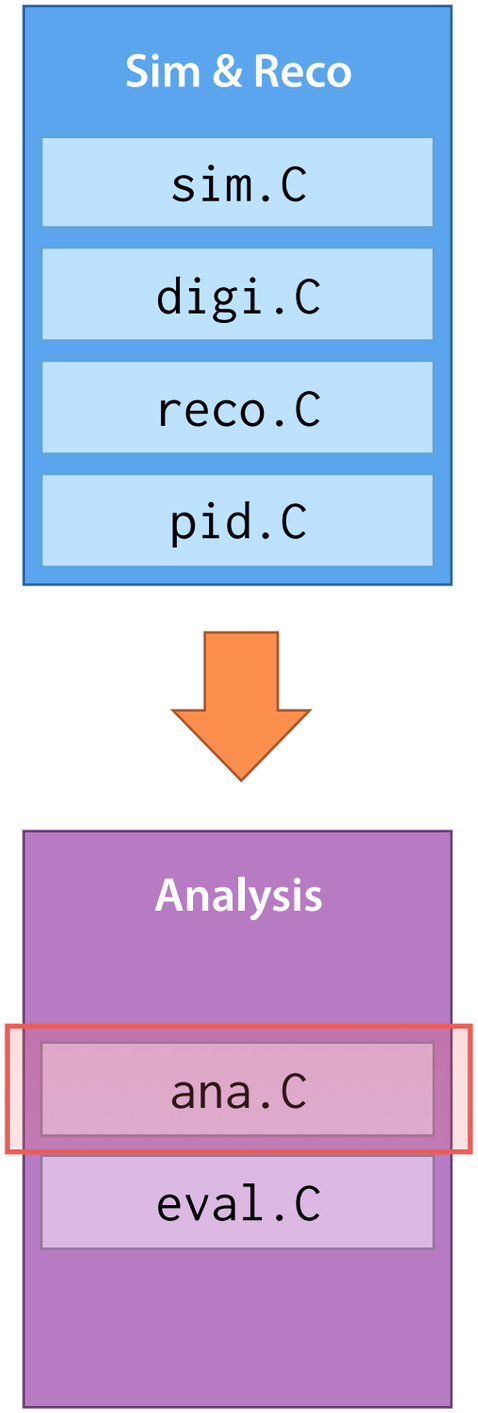
Analysis

- Concept
  - `ana.C` *(1+ file)*
    - » Uses PID info to reconstruct composite particles
    - » Calls fitters
    - » Fill all information into `TTrees`
    - » As few permanent cuts as possible
  - `eval.C` *(lots of files, one per topic: `evalMass.C`, ...)*
    - » Use `TTree` info from before, also with conditions
    - » Create beautiful histograms
- **ana runs once, `eval` runs a lot**



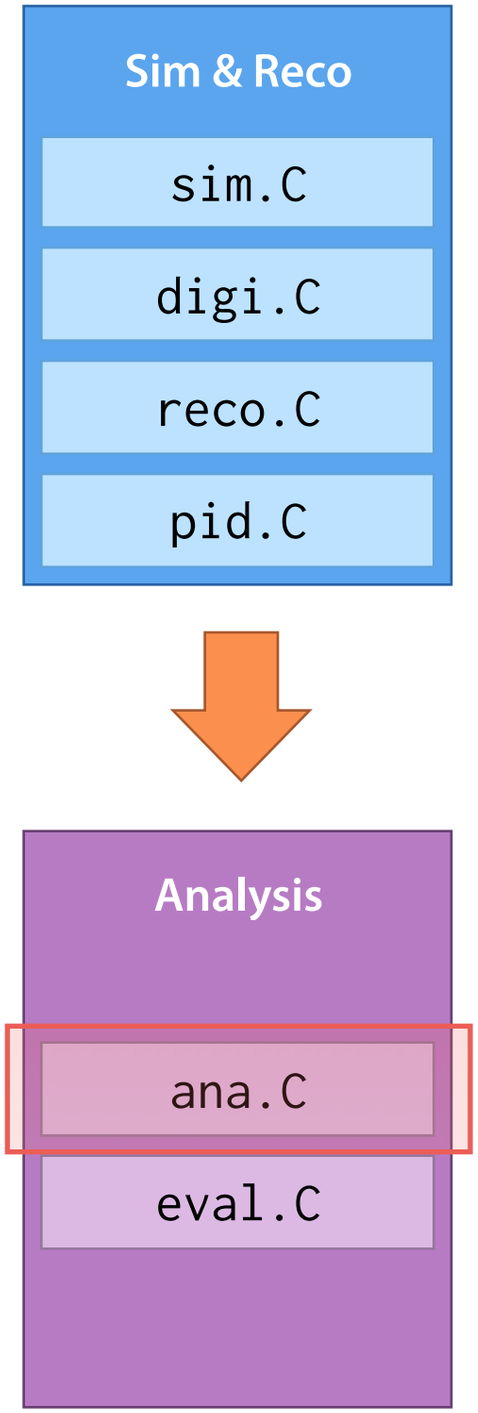
Analysis (ana.C)

- Analysis with Rho
- Rule of thumb: No histogram in ana.C
- Scheme
  - Create particles in list based on PID info
    RhoCandList is a collection of RhoCandidates
  - Extract information, save into TTree
  - Run fitters, apply cuts
  - Extract information, save into TTree
  - Combine particles (e.g. Kππ to D)
  - Extract info, run fitters, TTree…



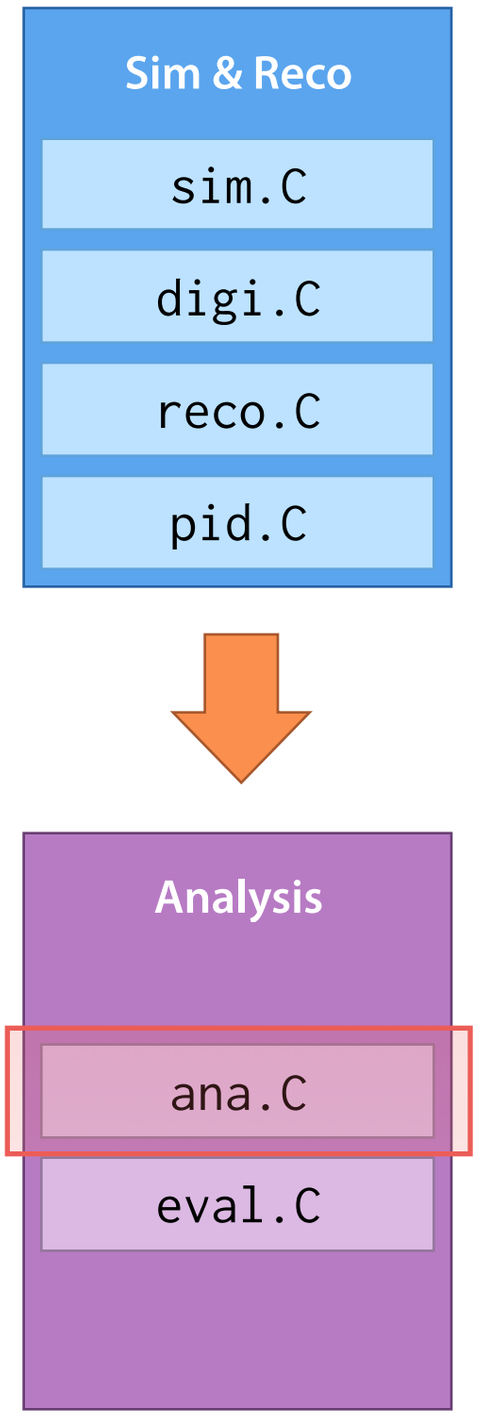
Analysis (ana.C) — Skeleton

```
void ana(TString prefix = "", int nevts = 0) {
    TFile * out = TFile::Open(anaOutput, "RECREATE");
    RhoTuple * ntpDp = new RhoTuple("ntpDp", "Dp Analysis");
    PndAnalysis* theAnalysis = new PndAnalysis();
    nevts = theAnalysis->GetEntries();

    RhoCandList kplus, piminus, dpluslist;
    double m0_dplus = TDatabasePDG::Instance()->GetParticle("D+")->Mass();
    RhoMassParticleSelector dMassSel("dp", m0_dplus, 0.3);
    TString pidSelection = "PidAlgoIdealCharged";
    PndRhoTupleQA qa(theAnalysis, pbarmom);

    while (theAnalysis->GetEvent() && i++ < nevts) {
        theAnalysis->FillList(kplus,  "KaonBestPlus", pidSelection);
        theAnalysis->FillList(piminus, "PionBestMinus", pidSelection);

        dpluslist.Combine(kminus, piplus, piplus);
        dpluslist.Select(&dMassSel);
        dpluslist.SetType(411);

        for (j = 0; j < dpluslist.GetLength(); ++j) {
            ntpDp->Column("ncandDplusI",  (Int_t) dpluslist.GetLength());
            // Next stuff goes here
            ntpDp->DumpData();
        }
    }
    out->cd();
    ntpDp->GetInternalTree()->Write();
    out->Save();
}
```



Analysis (ana.C) — Rho & QA

- Accessors of `RhoCandidate * cand`
  - `cand->GetMass()`
  - `cand->GetMomentum()`
  - `cand->GetPosition()`
  - `cand->P4()`
  - `cand->PdgCode()`
  - `cand->Cov7()` (i=j=0..2: σpos; i=j=3..6: σ4vec)
  - `cand->Boost(TVector3(1,2,3))`
  - `cand->NDaughter(), cand->Daughter(0)`
  - `cand->Lock()`
  - `cand->GetMcTruth()`
- Easier: QA tools!



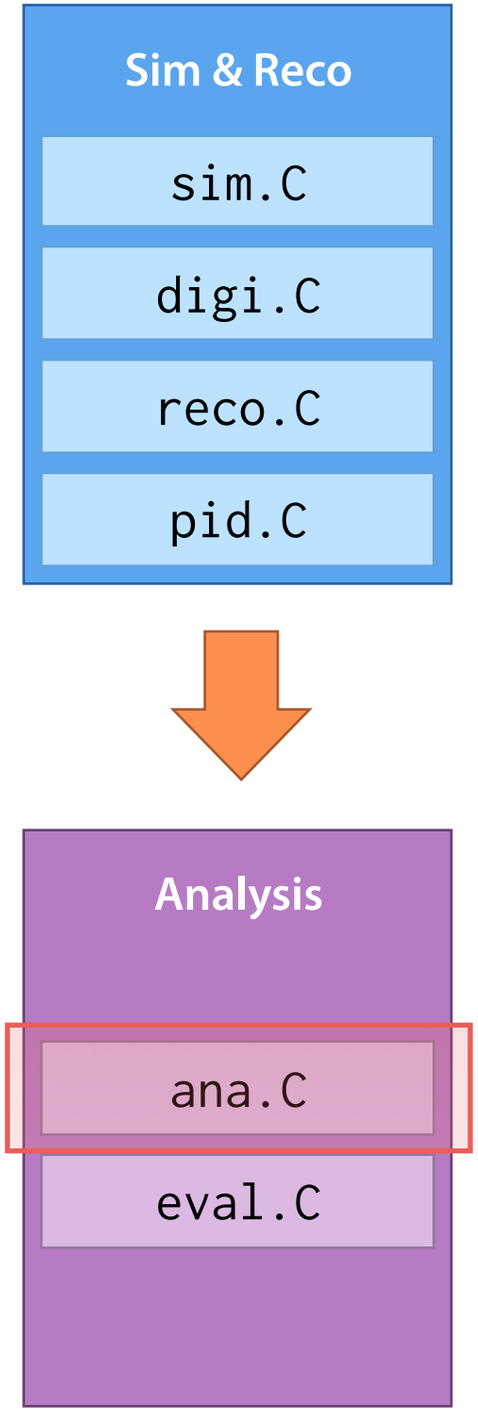
Analysis (ana.C) — Rho/**QA** General

```
qa.qaP4("beam", ini, ntpDp);
qa.qaCand("DplusPos", dpluslist[j], ntpDp);
qa.qaComp("Dplus", dpluslist[j], ntpDp);
qa.qaMcDiff("fVtxDplus", dplusfit, ntpDp);

RhoCandidate *truth = dpluslist[j]->GetMcTruth();
TLorentzVector lv;
if (0x0 != truth) lv = truth->P4();
qa.qaP4("trDplus", lv, ntpDp);
if (truth != 0x) {
    qa.qaVtx("trDplus", truth, ntpDp);
}

PndKinVtxFitter vertexFitter(dpluslist[j]);
vertexFitter.Fit();
RhoCandidate * dplusfit = dpluslist[j]->GetFit();
qaFitter("fVtxDplus", ntpDp, &vertexFitter);
ntpDp->Column("fVtxHowGood", (Int_t) dP_vtx_indexOfBestFit[j]);
qa.qaMcDiff("fVtxDplus", dplusfit, ntpDp);
```

1

2

3



Analysis (ana.C) — Rho/QA General 1

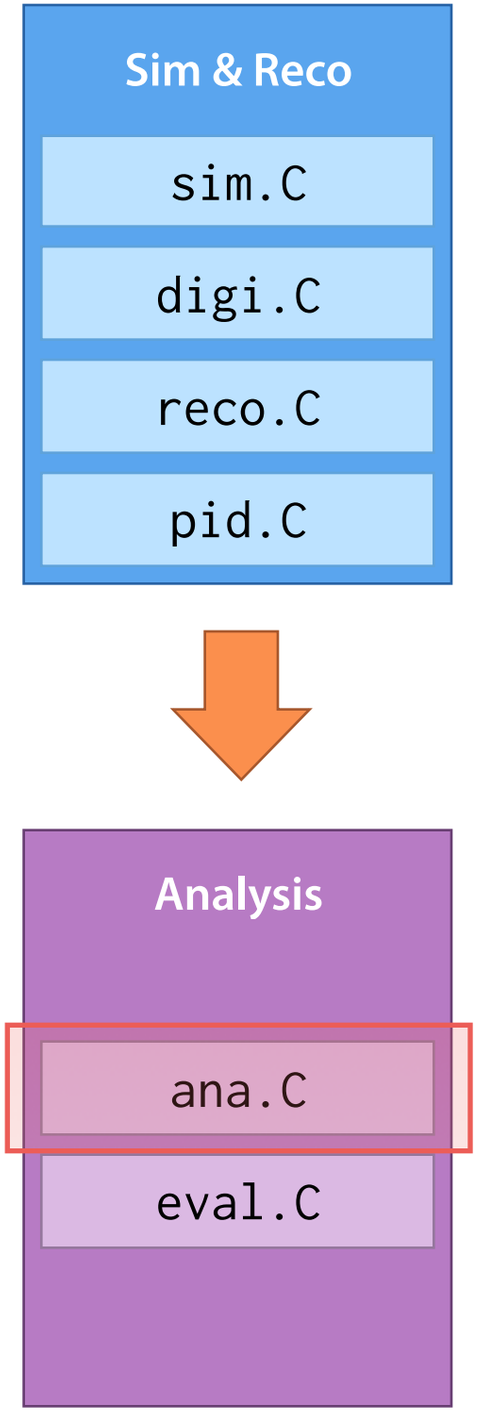
```
qa.qaP4("beam", ini, ntpDp);
qa.qaCand("DplusPos", dpluslist[j], ntpDp);
qa.qaComp("Dplus", dpluslist[j], ntpDp);
qa.qaMcDiff("fVtxDplus", dplusfit, ntpDp);
```

- **qaP4**: Stores four-momentum
  - p_i, p, E, theta, phi, m
  - qaP4Cms also existing
- **qaCand**: **qaP4** and spatial info
  - charge
  - x, y, z, l
  - PDG I
- **qaComp**: [illegible] above, plus more
  - For composite candidates
  - Runs recursively down the particle tree
  - Calls also: **qaPoca**, **qaVtx**
- **qaMcDiff**: Differences to MC
  - Δx, Δp, ΔE
  - $pull_x$, $pull_p$, $pull_E$
  - σ_x, σ_p, σ_E
  - Make cross-check!

That's the one you need!

- **qaKs0, qaPi0**
- **qaEventShape**
- **qaDalitz**
- **qaPid**
- **qaTrk**
- **qaEmc, qaMvd, qaStt, qaGem, qaDrc, qaDsc, qaRich, qaTof, qaMuo (= qaRecoFull; qaRecoFullTree)**
- **qaMc**
- **qaFitter**
- **qaPoca, qaPRG**

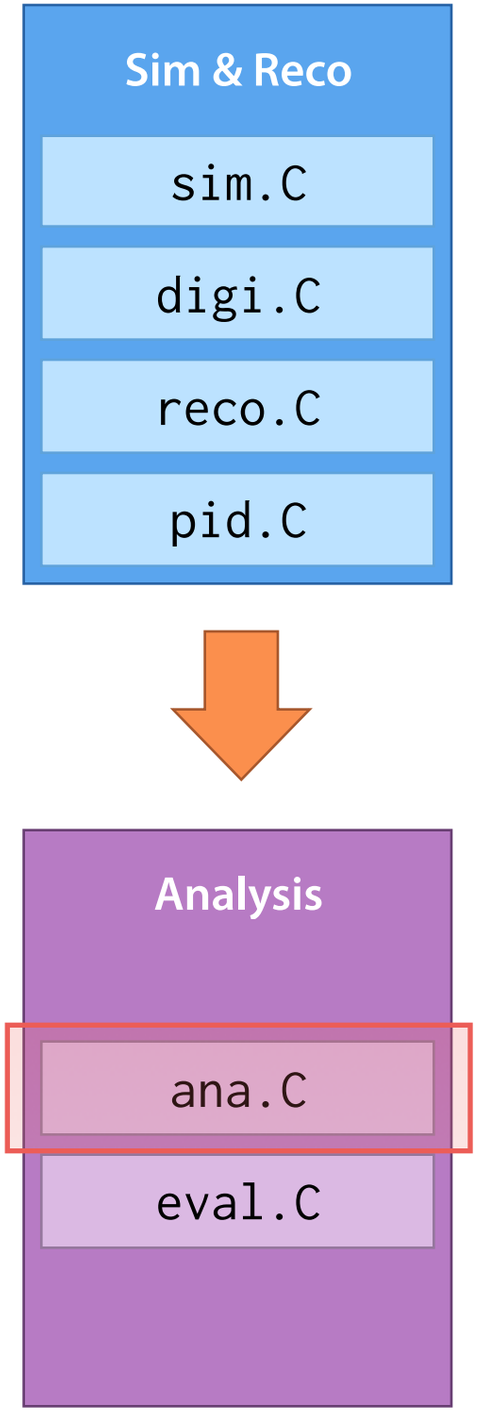
https://subversion.gsi.de/trac/fairroot/browser/pandaroot/trunk/PndTools/AnalysisTools/PndRhoTupleQA.cxx



Analysis (ana.C) — Rho/QA General 2

```
RhoCandidate *truth = dpluslist[j]->GetMcTruth();
TLorentzVector lv;
if (0x0 != truth) lv = truth->P4();
qa.qaP4("trDplus", lv, ntpDp);
if (truth) {
    qa.qaVtx("trDplus", truth, ntpDp);
}
```

- **dpluslist[j]->GetMcTruth()**: You can always get the truth!
  - Sometimes it fails (NULL pointer)
  - Check it!
    `truth != 0, 0x0, NULL, nullptr` or event just `if (truth)` – all are identical in C++
- **qaP4**(truth): I'd like to save the truth as well
- **qaVtx**: Saves vertex information
  - If `DecayVtx() != (0,0,0)`: Decay vertex (from fitter)
  - If `DecayVtx() == (0,0,0)`: Position of 4-vec of first daughter



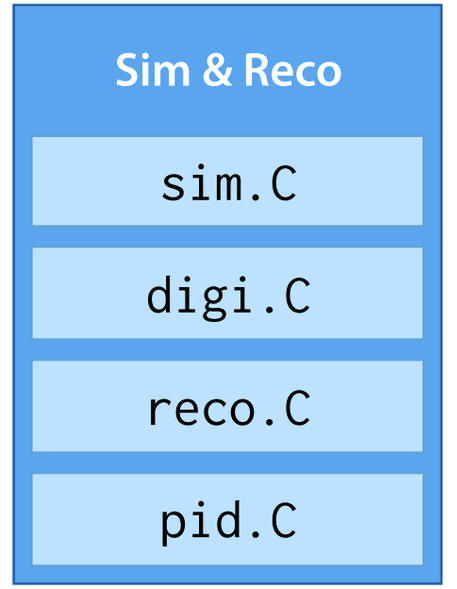
Analysis (ana.C) — Rho/QA Fitter


```
PndKinVtxFitter vertexFitter(dpluslist[j]);
vertexFitter.Fit();
RhoCandidate * dplusfit = dpluslist[j]->GetFit();
qaFitter("fVtxDplus", ntpDp, &vertexFitter);
ntpDp->Column("fVtxHowGood", (Int_t) dP_vtx_indexOfBestFit[j]);
qa.qaMcDiff("fVtxDplus", dplusfit, ntpDp);
```

- **Fitting**
  Fitter always take a RhoCandidate and save into that RhoCandidate
  – Usual workflow:
    `fitter(cand); fitter.fit(); cand->GetFit();`
  – The resulting RhoCandidate can be used usually
- Currently available fitters:

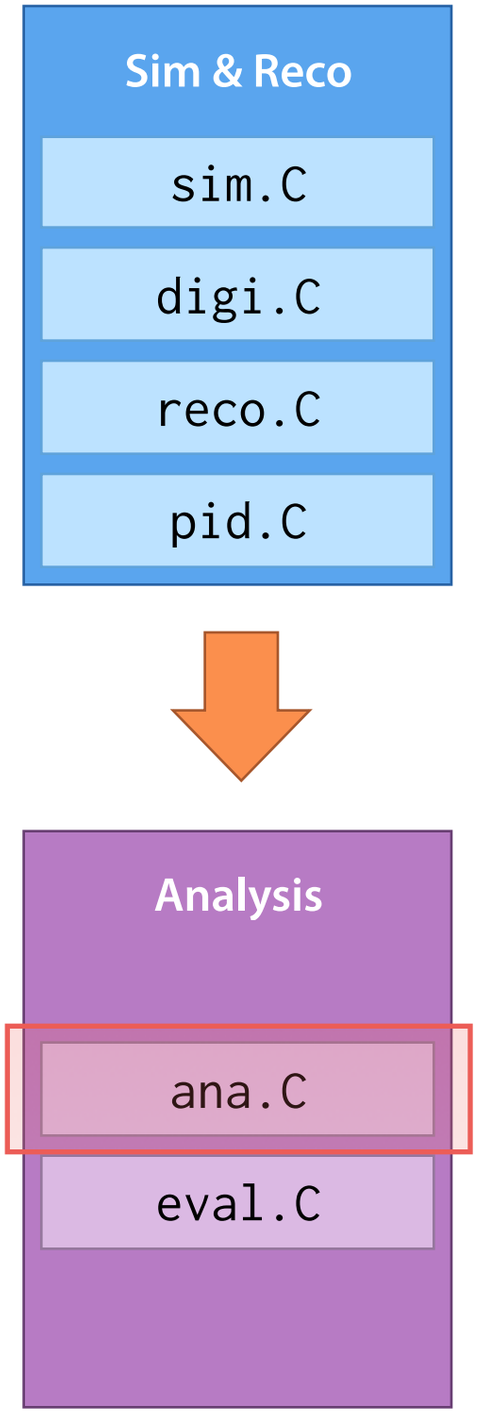
  https://panda-wiki.gsi.de/foswiki/bin/view/Computing/PandaRootRhoTutorial#A_2.5._Fitting

  https://subversion.gsi.de/trac/fairroot/browser/pandaroot/trunk/rho/RhoBase/RhoFitterBase.h
  – Vertex fitters (x_i)
    » `PndKalmanVtxFitter` *(faster)*
    » `PndKinVtxFitter` *(more precise)*
  – Four-Constraint (p_i, E)
    `Pnd4CFitter`
  – Mass-Constraint (m)
    `PndKinFitter` + `AddMassConstraint()`



Analysis (ana.C) — Rho/QA Fitter

- What if there's > 1 composite cand / evt?
  `dPluslist.length() > 1`
- Tag them!
- Our strategy
  - Do fits before filling TTrees
  - For every composite candidate, store how good fit is (= tag)
    - » Sort candidates by their fit χ^2 value
    - » Tag candidate with least χ^2 as 1, …
    - » (only, if Prob(χ^2) > 0.01; if not, -1, -2, …)
- Benefit of tagging: Always all information saved! Cross checks possible
- Tag accessible from TBrowser: `"fVtxHowGood == 1"`
- See André's development folder: `PndRhoFitProbPresorter`

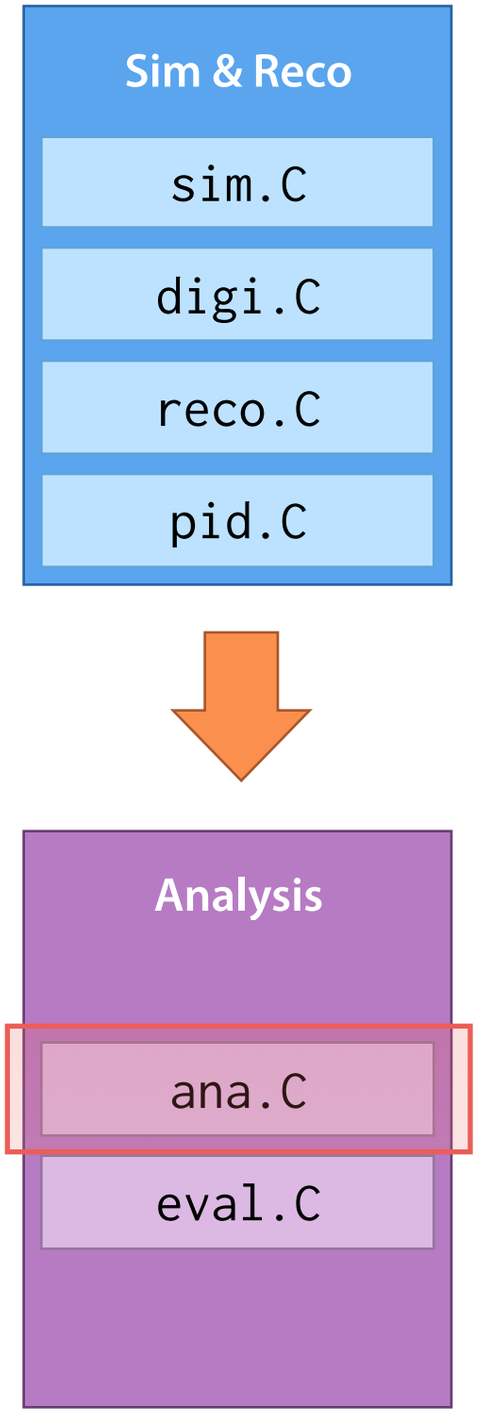


Analysis (ana.C) — Rho Tips'n'Tricks

- Never create your own RhoCandidates

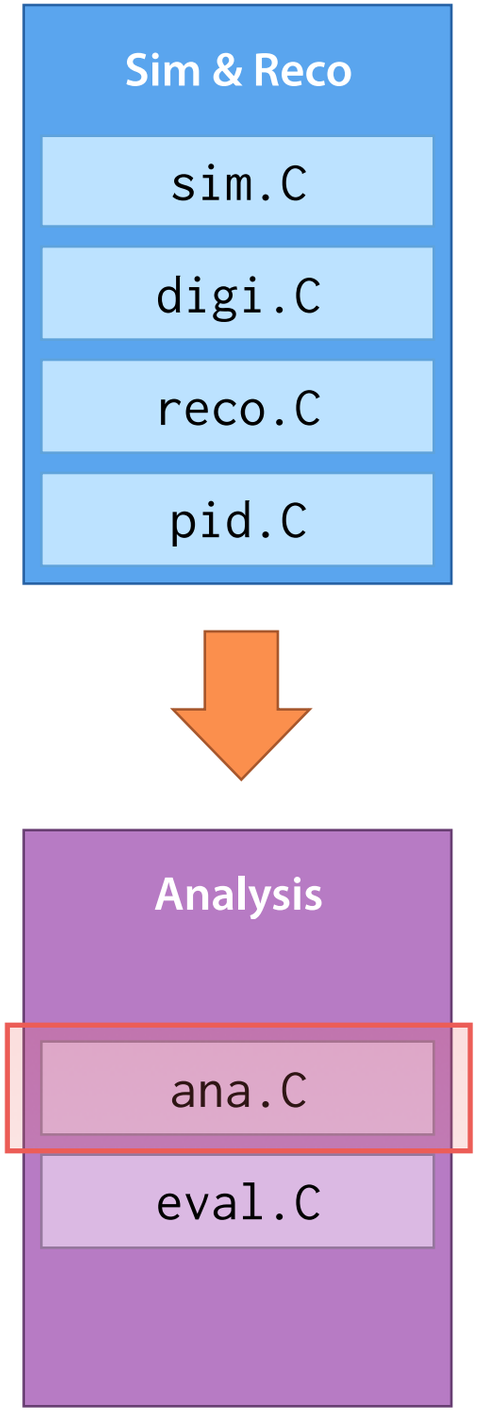
```
RhoCandidate *c = RhoFactory::Instance()->NewCandidate();
```

- Comparing RhoCandidates for equality
  - `c1 == c2`
  - `c1->Equals(c2)`
  - `c1->GetMarker() == c2->GetMarker()`
- RhoCandList
  - `Append(), Combine(), CombineAndAppend()`
  - `Overlaps(), RemoveFamily(), RemoveClones()`
- Use QA, but check!
  - Write own qaExtensions *(next slide)*



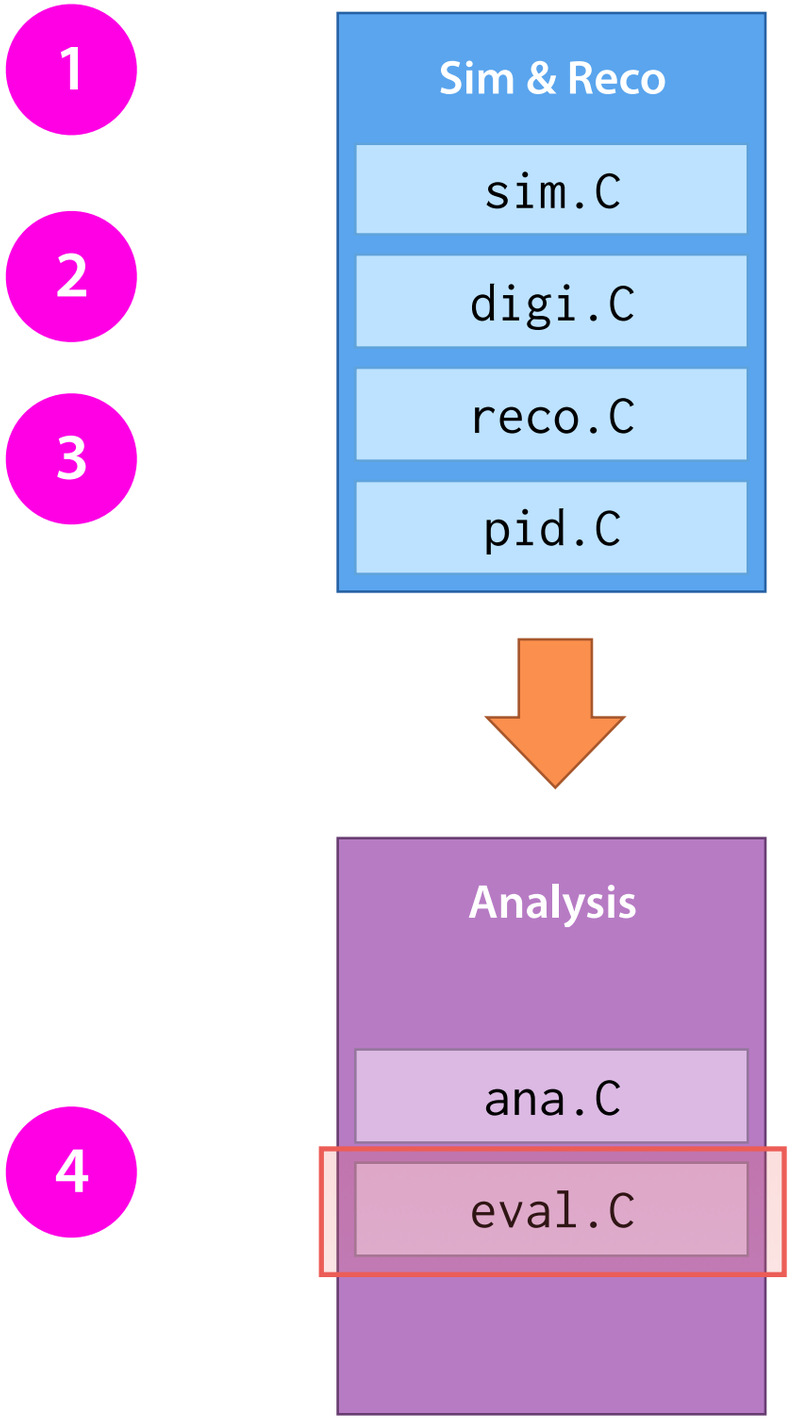
Analysis (ana.C) — Rho Extensions

```
namespace andi {
    void qaFitter(PndRhoTupleQA * qa, TString pre, RhoCandidate * c,
RhoTuple * n, RhoFitterBase * fit, bool skip = false) {
        if (0 == n) return;

        if (!skip) {
            qa->qaComp(pre, c, n);
            n->Column(pre + "Chi2", (float) fit->GetChi2(), 0.0f);
            n->Column(pre + »Ndf",  (float) fit->GetNdf(),  0.0f);
            n->Column(pre + "Prob", (float) fit->GetProb(), 0.0f);
        }
    }
    void qaDaughters(PndRhoTupleQA * qa, TString pre, RhoCandidate * c,
RhoTuple * n) {
        if (0 == n) return;
        for (int i = 0; i < c->NDaughters(); ++i) {
            RhoCandidate * daughter = c->Daughter(i);
            RhoCandidate * daughterTrue = daughter->GetMcTruth();
            TString dindex = TString::Format("d%i", i);
            qa->qaCand(pre + dindex + "tr", daughterTrue, n);
            n->Column(pre + dindex + "trpdg", (Int_t) daughterTrue-
>PdgCode());
            qa->qaMcDiff(pre + dindex, daughter, n);
        }
    }
}
```



Evaluation (eval.C)

- Read in TTree with QA tuples (or TTrees)
- Look at quantities in TBrowser
- Create histograms
  - Use TTrees as much as possible
  - SetBranchAddress() is more complicated and usually slower
  - See http://static.andreasherten.de/2014/06/23/ROOT-NTuple-Analysis.html
- Use helper functions

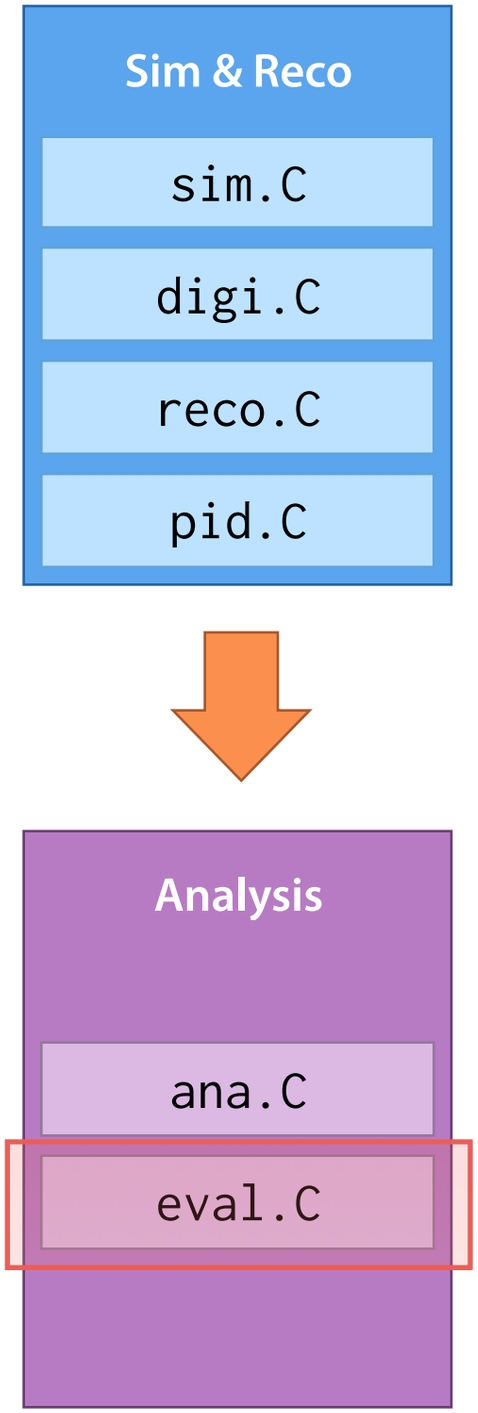


Evaluation (`eval.C`) — Skeleton

1

```
void evalAna(TString inputFile = "output_ana.root") {
    andi::setCustomStyle();
    bool savePictures = true;
    TString macroBasename = "evalAna.C";

    TFile * file = new TFile(inputFile);
    TTree * tupleDplus = (TTree*)(file->Get("ntpDp"));
    // ... do things
}
```

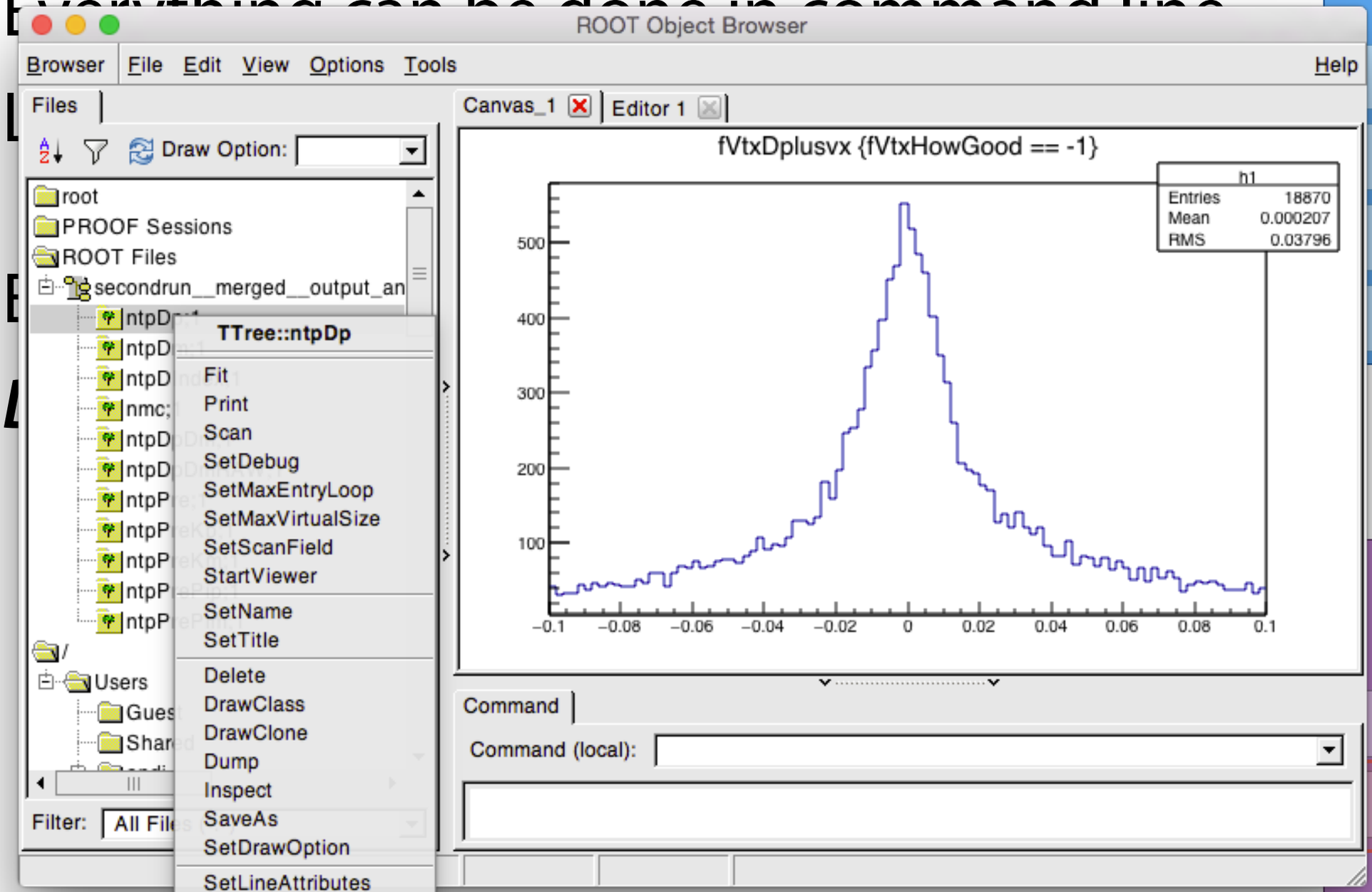
- Things
  - `setCustomStyle`
    Sets margins and font sizes and other default parameters
  - `savePictures`
    Global flag for macro which automatically saves all created histograms (or not)
  - `macroBasename`
    The name of the current macro; used mainly for saving images

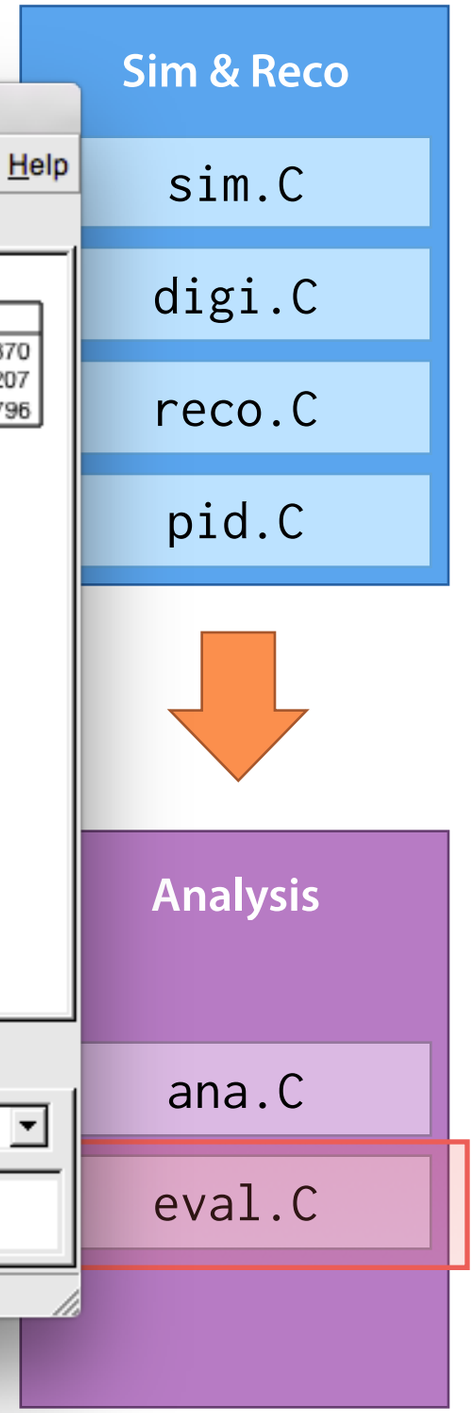


Evaluation (eval.C) — TBrowser

2

- Since ROOT has an interpreter:
  Everything can be done in command line
- [illegible]
- E[illegible]
- L[illegible]

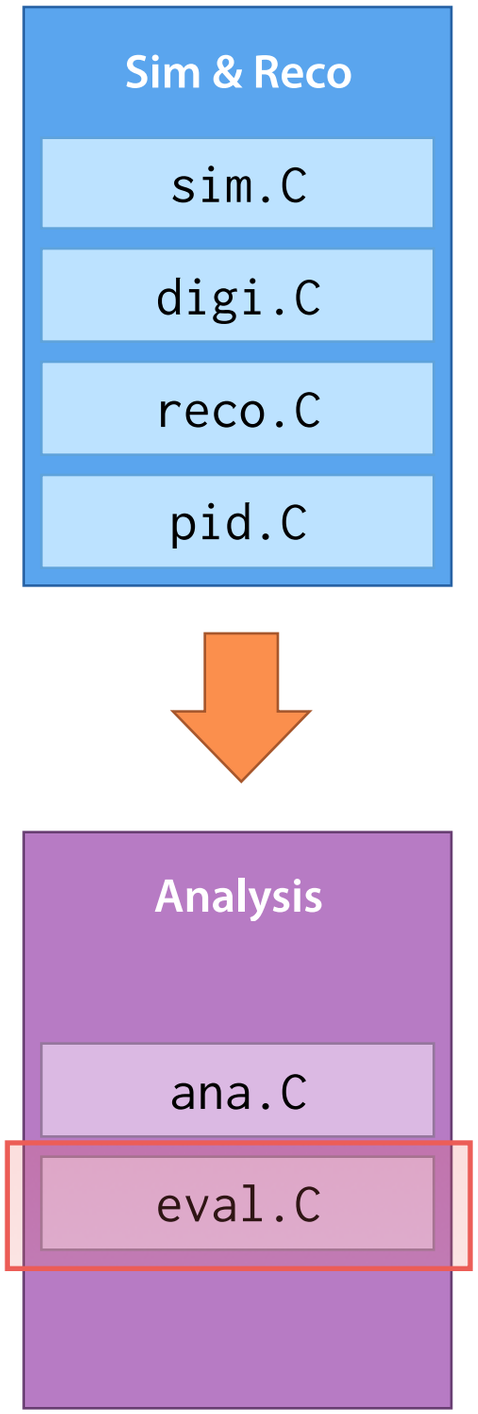




Evaluation (eval.C) — TBrowser II 2

- Not actually in eval.C, but interactively in ROOT command line

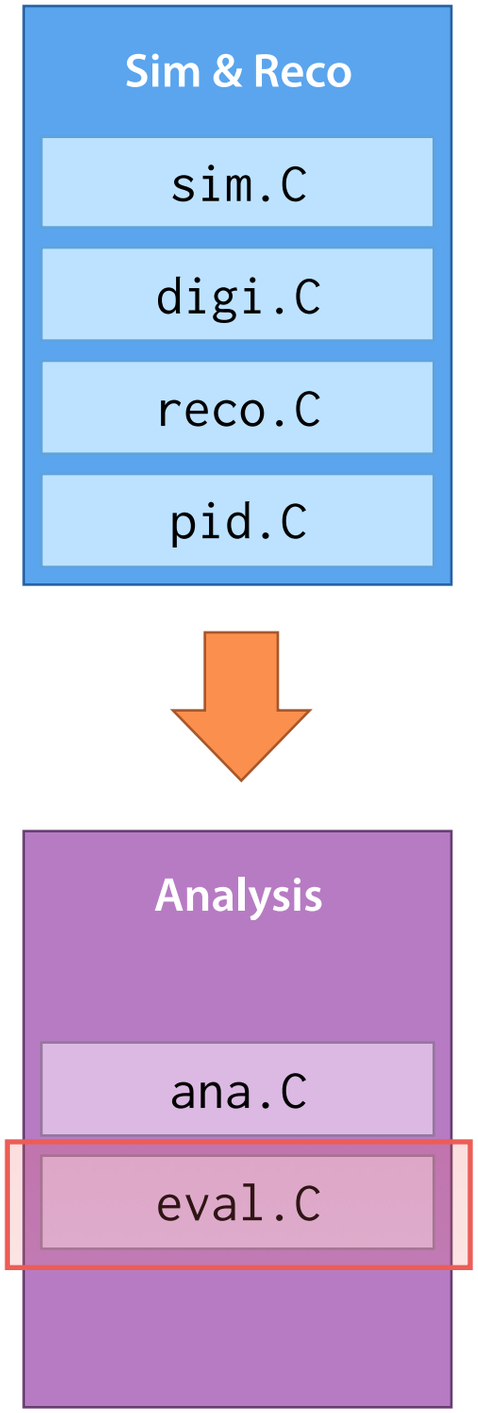
```
tv__tree->Draw("fVtxDplusvx");

tv__tree->Draw("fVtxDplusvx", "fVtxHowGood == 1 && abs(Dplusd1x) > 0.2");

TCut default = "fVtxHowGood == 1";
tv__tree->Draw("fVtxDplusvx", default && "abs(Dplusd1x) > 0.2");

tv__tree->Draw("sqrt(fVtxDplusvx^2 + fVtxDplusvy**2)", default);

TH1D h1("h1", "Temp", 100, 0, 1);
tv__tree->Draw("sqrt(fVtxDplusvx^2 + fVtxDplusvy^2)>>h1", default);

tv__tree->Draw("sqrt(fVtxDplusvx^2 + fVtxDplusvy^2)>>h2(100, 0, 1)",
default);

tv__tree->Draw("fVtxDplusvx^2:fVtxDplusvy^2>>h3(100, -1, 1, 100, -1, 1)",
default, "COLz");


tv__tree->Scan("evt:fVtxDplusvx^2:fVtxDplusvy^2", default);
```



Evaluation (`eval.C`) — Histograms 3

- In `eval.C`: Do as in ROOT command line, but `Project()` instead of `Draw()`

```
void evalAna(TString inputFile = "output_ana.root") {
    andi::setCustomStyle();
    bool savePictures = true;
    TString macroBasename = "evalAna.C";

    TFile * file = new TFile(inputFile);
    TTree * tupleDplus = (TTree*)(file->Get("ntpDp"));

    TCut cutBoth = "fVtxHowGood == 1 && fMassHowGood == 1";

    TH2D * hDp_ptpz = new TH2D("hDp_ptpz", "D^{+}: Momentum
Distribution;p_{z} / GeV/c;p_{t} / GeV/c", 100, 1, 5, 100, 0, 1);
    tupleDplus->Project("hDp_ptpz", "Dpluspt:Dpluspz", cutBoth);
    hDp_ptpz->SetLineColor(kBlue);
    hDp_ptpz->SetTitleOffset(1.01);
    hDp_ptpz->Draw("COLz");
}
```



Evaluation (eval.C) — Helpers

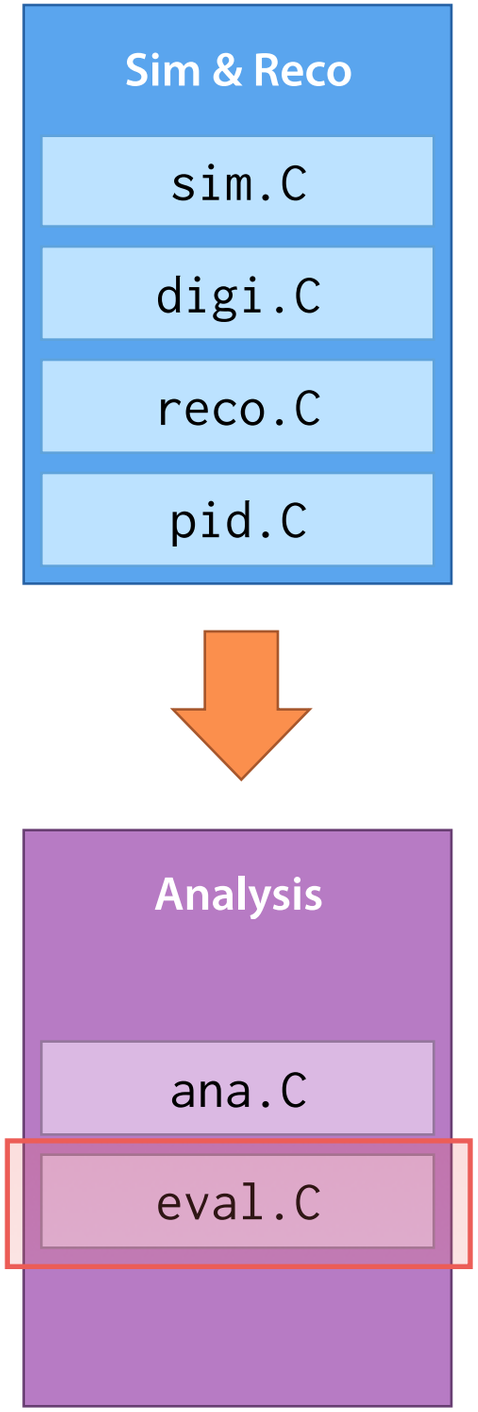
- My workflow for simple plots (95 %)

```
TH1D * hDp_m = new TH1D("hDp_m", "D^{+}: Invariant Mass;m / GeV/
c^{2};counts", 100, 1.7, 2.05);

hDp_m->SetLineColor(colorDp);

tupleDplus->Project("hDp_m", "Dplusm");

andi::createCanvasDrawAndSave(hDp_m, "D Plus Mass", macroBasename,
savePictures);
andi::createCanvasFitDoubleDrawAndSave(hDp_m, "D Plus Mass Relative",
macroBasename, savePictures);
```

- ***Demo***
- common.cpp

  https://github.com/AndiH/PhD/blob/master/Programming/common.cpp

  Documentation at http://andih.github.io/PhD/



Helpers: common.cpp

- common.cpp: my file for helper functions

  https://github.com/AndiH/PhD/blob/master/Programming/common.cpp

- Loaded into ROOT via ~/.rootrc with

  ```
  Rint.Load: ~/Documents/Coding/PhysicsAnalysis/common.cpp
  ```

  Available globally, also in command line!

- Name-spaced functions and classes

- Documentation included, generate and open with

  ```
  $ doxygen common.cpp
  $ open html/index.html
  ```

  Or simply look at http://andih.github.io/PhD/

- Since we're at it…

  ```
  alias r="root -l"
  ```

  https://github.com/AndiH/PhD/blob/master/Programming/root-aliases.sh

Helpers: common.cpp

- moveStatBoxLeft(), moveStatBoxDown(), shrinkBox()
- makePadTitle()
- saveCanvas()
- histogramsToStack()
- gaussFit(), doubleGaussFit()
- createCanvasDrawAndSave()
- treeFromMultipleFiles() *(see later)*
- DInfoContainer *(see next)*

Helpers: `DInfoContainer`

- Sometimes, `Draw()` / `Project()` is not possible (x-relations, …)

  → `SetBranchAddress()` needed

- A lot of quantities → *messy* code
- Most common particle properties: m, E, p_i, …
- Idea: create `struct` to simplify

```
struct properties {
    Float_t pt, px, py, pz, p, E;
    Float_t m, chg, pdg;
};
struct DInfoContainer {
    properties m; // actual d meson (m=mother)
    properties d0, d1, d2; // three daughters
};
DInfoContainer container;
tuple->SetBranchAddress("Dpluspt", &(container.m.pt));
// …
```

Prometheus – Login – SSH 1

- Login via **SSH key**
  - Faster, more reliable
  - No need to remember password

1. Generate SSH key:

```
$ ssh-keygen -t rsa -b 4096 -C "a.herten@fz-juelich.de"
```

2. Copy SSH key to remote location;

```
$ cat ~/.ssh/id_rsa.pub | ssh user@hostname 'cat
>> .ssh/authorized_keys'
```

OR: `$ ssh-copy-id user@hostname`

3. Login:

```
$ ssh user@hostname
```

Prometheus – Login – SSH 2

- **Shortcuts** for hosts (`~/.ssh/config`)

```
Host ikp
    User herten
    HostName ikpwsg1-sshg.ikp.kfa-juelich.de
    ForwardX11 yes

Host ikp507
    User herten
    ProxyCommand ssh -q ikp netcat ikp507 22
    ForwardX11 yes


Host gsi
    User aherten
    HostName lxpool.gsi.de
    ForwardX11 yes

Host prometheus
    User aherten
    ProxyCommand ssh -q gsi netcat pro.hpc 22
    ForwardX11 yes
```

Prometheus – Copy

- Two means of copy
  - scp:

```
$ scp prometheus:/hera/panda/aherten/file.root .
```

  - rsync:

```
$ rsync -v --partial --progress --human-readable
    --rsh=ssh prometheus:/hera/panda/aherten/data/
    file.root .
```

  - Important `rsync` option for backupping your files:

```
$ rsync -a
```

Prometheus – Job Submission

- Documentation
  - Main documentation (GSI):

    https://wiki.gsi.de/cgi-bin/view/Linux/GridEngine
  - PandaRoot-oriented how-to (Klaus):

    https://panda-wiki.gsi.de/foswiki/bin/view/Computing/PandaRootSimulationPrometheus
- qsub for submitting,
  qstat for statistics
- Get a submission script from us, look at manpage:

  ```
  $ man qsub
  ```
- Usually:

  ```
  $ qsub -t 1-100 signalData.sge initialProd 2000
  ```
- Status:

  ```
  $ watch qstat
  ```

Prometheus – Job Resubmission

- Use bash to your advantage
- Find crashed jobs:

```
$ find initialProd_*_pid.root -type f -size -2M > failed_jobs.txt
```

- Extract job numbers from crashed jobs:

```
$ cat failed_jobs.txt | cut -d "_" -f 2 > failed_ids.txt
```

- Resubmit crashed job numbers:

```
# resubmit.sh
while read i; do
  qsub -t $i signalData.sge initialProd 2000
done < failed_ids.txt
```

- Random Bash-Fu:

```
cp {a,b}.root ../   ⇔   cp a.root ../; cp b.root ../
cp {1..3}.root ../  ⇔   cp 1.root ../; cp 2.root ../; cp 3.root ../
```

Prometheus – File Merging

- My concept of analysis (`ana.C`/`eval.C`)
  - `{sim,digi,reco,pid}.C` run on Prometheus → `pid_`*n*`_complete.root`
  - `ana.C` runs on Prometheus → `ana_`*n*`_complete.root`
  - `eval.C` runs locally → `histogram.pdf`
- Merging of `ana_`*n*`_complete.root` on Prometheus
- Easiest way: `hadd`

```
$ hadd -h
Usage: hadd [-f[0-9]] [-k] [-T] [-O] [-n maxopenedfiles] [-v verbosity] targetfile source1
[source2 source3 ...]
This program will add histograms from a list of root files and write them
to a target root file. The target file is newly created and must not
exist, or if -f ("force") is given, must not be one of the source files.
```

- Merge only *dumb* ROOT files with no cross-references,
  - OK: Histograms
  - OK: Rho TTrees
  - NOT OK: PID files

Prometheus – File Merging II

- Easiest way: `hadd`

```
$ hadd output.root ana_{1..100}_complete.root
```

- Problem: everything goes to memory!
  Not feasible for a lot of and/or large files
- 2nd order easiest way: `hadd` loops!

  → Becomes cumbersome very quickly
- `mergeRootFilesByTxt.sh`!

  https://gist.github.com/AndiH/121e3d89fc647fb65338
  - Input: a `.txt` file with a name of one `.root` file per line
  - Output: a merged `.root` file
  - Usage: `./mergeRootFilesByTxt.sh -h`

Prometheus – File Merging III

- For me, `hadd` made problems for `.root` files with unequally long `TTrees` inside *(it filled the shorter `TTree` with 0s)*
- Solution: `andi::treeFromMultipleFiles()`

```
TTree * tupleDplus = andi::treeFromMultipleFiles("Dplus", "files.txt")
```

  – Quite slow!

- Minimal example:

```
TTree * tupleDplus = andi::treeFromMultipleFiles("Dplus", "files.txt")
TFile * f = new TFile("output.root", "RECREATE");
TTree * clone = tupleDplus->CloneTree();
if (clone) clone->Write("", TObject::kOverwrite);
f->Close();
```

Klaus' Tools

- `fitsb.C`: Fit signal+background distribution

  https://subversion.gsi.de/trac/fairroot/browser/pandaroot/trunk/macro/analysistools/fitsb.C

- `findcuts.C`: Finds best cut criteria for *S* and *B*

  https://subversion.gsi.de/trac/fairroot/browser/pandaroot/trunk/macro/analysistools/findcuts.C

  – Must be both in same `TTree`

  – Works interactively

- More at `macro/analysistools/`

JÜLICH
FORSCHUNGSZENTRUM

panda

THE END