An analysis in ALICE: from data taking, to publication

Michal Broz

Beginning

- You or some kind of your supervisor will find a **doable** analysis and you can start working on it
    - To find a problem which can lead to an interesting result and a solution can be found in a meaningful time is highly non-trivial
- Usually you begin with developing an code to analyze the data and produce first partial results
- Once you have some kind of plots you present them at the level of Physics-Analysis-Group (PAG) to let people know that you are actually working on it
- You request an talk on the PAG meeting every here and there (two weeks is a good basic interval) to show a progress

Analysis progress

- PAG is a group of people interested in the field and most probably contributing to another analysis in the similar topic either by themselves or exploiting a student-slave
- Often some PAG members are pure theoreticians, usually authors of the predictions you will use at some point
- PAG thus can provide you a support and sometimes a valuable comments
- But the responsible person for the analysis is still you
- PAG cannot really substitute your supervisor :)

Analysis in progress

- You may be working on some kind of pre-processed data provided by higher instance
- But that not optimal solution you and some point you would need to switch to your own code and run over real data

- There is bunch of detailed presentations and tutorials on many topics here:
- http://alice-analysis.web.cern.ch

- Try yourself on sunrise.fjfi.cvut.cz !

Analysis in progress

- Login to sunrise.fjfi.cvut.cz
- Go to /user_data/alice/NanoAODs
- There:
  - Data: LHC15o_Runs
  - Code: AnalysisSkeleton
- Copy the skeleton to your home, edit, run
- Use chunks of data to test or run over all of them via the batch system
- Scripts for use of batch are in skeleton folder

All analysis task are derived from **AliAnalysisTaskSE**

- This provides us with a '**common framework**' in which to do analysis (i.e. all analyses have the same **base** methods)

```
AliAnalysisTaskSE::AliAnalysisTaskSE();
AliAnalysisTaskSE::AliAnalysisTaskSE(const char*);
AliAnalysisTaskSE::~AliAnalysisTaskSE();
AliAnalysisTaskSE::UserCreateOutputObjects();    // histos
AliAnalysisTaskSE::UserExec(Option_t*);          // anaysis
AliAnalysisTaskSE::Terminate(Option_t*);         // finish
```

We'll use a **fixed format** for writing our class

- The **header** file (.h) which contains function prototypes
- The **implementation** file (.cxx), in which the code is implemented
- An **AddTask.C** macro which configures the analysis

Let's start by looking at our header, as here all functions are defined

```cpp
#ifndef AliAnalysisTaskMyTask_H
#define AliAnalysisTaskMyTask_H
class AliAnalysisTaskMyTask : public AliAnalysisTaskSE
{
public:
   // two class constructors
   AliAnalysisTaskMyTask();
   AliAnalysisTaskMyTask(const char *name);
   // class destructor
   virtual                 ~AliAnalysisTaskMyTask();
   // called once at beginning or runtime
   virtual void            UserCreateOutputObjects();
   // called for each event
   virtual void            UserExec(Option_t* option);
   // called at end of analysis
   virtual void            Terminate(Option_t* option);
ClassDef(AliAnalysisTaskMyTask, 1);
};

#endif
```

Let's start filling in the blanks, and add a histogram, an output list and some more

- Here's our updated **header** file

```
...
class AliAnalysisTaskMyTask : public AliAnalysisTaskSE
...
 private:
   AliAODEvent*  fAOD;             //! input event
   TList*        fOutputList;      //! output list
   TH1F*         fHistPt;          //! dummy histogram
...
```

- Pointers to objects that are initialized at run-time (e.g. the output list and histograms) should have `//!` at the end
- We use to output list to have one common output object, rather than many

- ROOT requires **two** class constructors

```
AliAnalysisTaskMyTask::AliAnalysisTaskMyTask() :
    AliAnalysisTaskSE(),
    fAOD(0), fOutputList(0), fHistPt(0)
{
    // ROOT IO constructor, don't allocate memory here!
}
AliAnalysisTaskMyTask::AliAnalysisTaskMyTask(const char* name)
    : AliAnalysisTaskSE(name),
    fAOD(0), fOutputList(0), fHistPt(0)
{
    DefineInput(0, TChain::Class());
    DefineOutput(1, TList::Class());
}
```

- We'll get later to the why

Objects that are **output** are initialized in the function

- UserCreateOutputObjects()

```
...
#include "TList.h"
#include "TH1F.h"
...
AliAnalysisTaskMyTask::UserCreateOutputObjects()
{
    // create a new TList that OWNS its objects
    fOutputList = new TList();
    fOutputList->SetOwner(kTRUE);

    // create our histo and add it to the list
    fHistPt = new TH1F("fHistPt", "fHistPt", 100, 0, 100);
    fOutputList->Add(fHistPt);

    // add the list to our output file
    PostData(1,fOutputList); // recall 1 from constructor
}
```

The function **UserExec()** is called for each event

- This is the 'heart' of our analysis

```
        ...
    if(!fAOD) return;
    // let's loop over the trakcs and fill our histogram
    Int_t iTracks(fAOD->GetNumberOfTracks());
    for(Int_t i(0); i < iTracks; i++) {
        // loop over all the tracks
        AliAODTrack* track = static_cast<AliAODTrack*>(fAOD->
    GetTrack(i));
        if(!track) continue;
        // fill our histogram
        fHistPt->Fill(track->Pt());
    }
    // and save the data gathered in this iteration
    PostData(1, fOutputList);
}
```

In principe, this is all we need in the .cxx and .h

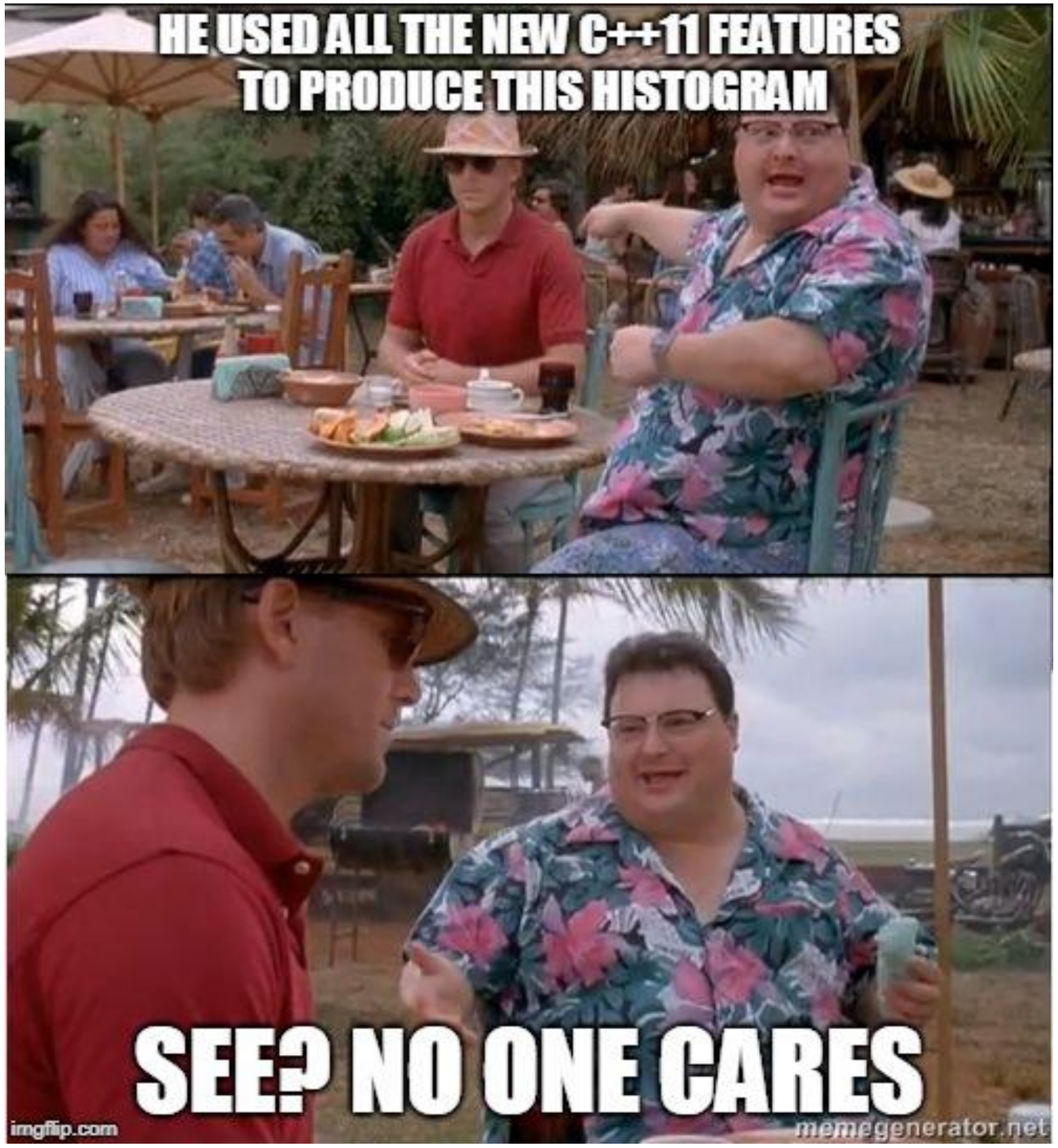
The **AddMyTask.C** macro instantiates our task (class), define it's in- and output and connect it to the analysis manager

```
AliAnalysisTaskMyTask* AddMyTask(TString name = "name") {
  AliAnalysisManager *mgr = AliAnalysisManager::
    GetAnalysisManager();

  TString fileName = AliAnalysisManager::GetCommonFileName();
  fileName += ":MyTask";       // create a subfolder in the file
  // now we create an instance of your task
  AliAnalysisTaskMyTask* task = new AliAnalysisTaskMyTask(name.
    Data());
  // add your task to the manager
  mgr->AddTask(task);
  // connect the manager to your task
  mgr->ConnectInput(task,0,mgr->GetCommonInputContainer());
  // same for the output
  mgr->ConnectOutput(task,1,mgr->CreateContainer("
    MyOutputContainer", TList::Class(), AliAnalysisManager::
    kOutputContainer, fileName.Data()));
  // important: return a pointer to your task
  return task;
}
```

```cpp
void runAnalysis() {
    // header location
    gROOT->ProcessLine(".include $ROOTSYS/include");
    gROOT->ProcessLine(".include $ALICE_ROOT/include");
    // create the analysis manager
    AliAnalysisManager *mgr = new AliAnalysisManager("
    AnalysisMyTask");
    AliAODInputHandler *aodH = new AliAODInputHandler();
    mgr->SetInputEventHandler(aodH);
    // compile the class (locally)
    gROOT->LoadMacro("AliAnalysisTaskMyTask.cxx++g");
    // load the addtask macro
    gROOT->LoadMacro("AddMyTask.C");
    // create an instance of your analysis task
    AliAnalysisTaskMyTask *task = AddMyTask();
    // if you want to run locally, we need to define some input
    TChain* chain = new TChain("aodTree");
    chain->Add("/scratch/.../AliAOD.root");
    // start the analysis locally
    mgr->StartAnalysis("local", chain);
}
```

- Try to make your code **as simple as possible**


HE USED ALL THE NEW C++11 FEATURES TO PRODUCE THIS HISTOGRAM
SEE? NO ONE CARES
imgflip.com
memegenerator.net

Preliminary

- It may take **years** to push the paper through the collaboration (more later)
- You want to show your results (figures) in advance on the conferences etc.
- At some point you will request figures to be approved as **Preliminary** by collaboration
- You don’t need approval to show something on CFRJS seminar or other local occasion
- You don’t need approval to show results in your thesis

Preliminary

- Physics Preliminary figures show the results of analysis and must include estimates of all statistical and systematic uncertainties in the underlying analysis that are relevant for the interpretation of the measurement and the understanding of underlying physics. There will be only one version of each preliminary result. Numerical values of preliminary results may be given to persons who are not members of the ALICE collaboration on request. Such requests are handled by Physics Coordination. Preliminary results are superseded by the published version of the results.
- Each ALICE Technical or Physics Preliminary figure must be presented at the Physics Forum and be approved by the PWG convener(s) and Physics Board before it can be shown outside the Collaboration

Route to publication

- Once an analysis is sufficiently advanced, an Analysis Note is prepared and presented to the PWG. This note contains all information needed for the reproduction of the analysis
- The PWG convener(s) and PAG coordinator(s) appoint Analysis Review Committees (ARC) whose task is to follow the analysis progress and the preparation of the Analysis Note critically and provide support and feedback to the people carrying out the analysis
- **Here is kink to Preliminary**
- The PWG convener(s) determine if a physics analysis is ready for consideration as a paper; i.e. the content of the paper is defined, the Analysis Note is approved

Route to publication

- The PWG convener(s) recommend the paper for the presentation at the Physics Forum and for the PB approval.
- Upon the PB approval the PWG convener(s) appoint the PC to prepare the initial manuscript and to create a dedicated page on the ALICE publication web site. The PB may require at this stage revisions or the merging of several ongoing analyses into a single paper.
- The Internal Review Committee (IRC) is appointed by the EB once a first complete draft of the paper is made available for review
- The IRC comprises experts and non-experts on the topic of the manuscript, drawn from across the Collaboration. The IRC carries out a comprehensive review of the physics analysis, accompanying documentation, and the text of the initial manuscript, as well as revisions to the manuscript

Route to publication

- First Collaboration Review
- The IRC reviews the manuscript and supporting documentation, and recommends corrections and changes as necessary.
- Upon approval of the draft by the IRC, the EB verifies that the actions of the PC and IRC meet the required standards, and reviews the draft before approving it for circulation to the full collaboration. The EB review is expected to be delivered within 5 working days.
- Upon EB approval, the EB circulates the draft to the full Collaboration for detailed comment for 10 working days.
- Up to 5 member institutes are specifically requested by the EB to comment in detail during the Collaboration review period.

Route to publication

- Second Collaboration Review •
- The PC prepares a new draft and a set of replies to the Collaboration comments.
- The IRC and the EB review the revised draft and responses to comments, and recommend relevant corrections and changes as necessary and appropriate.
- The PB is involved in case of major changes or open issues.
- Upon IRC approval, the EB circulates the revised manuscript, including revisions to the author list that arose, to the full collaboration for comments for a minimum of 5 working days
- The main purpose of this second comment period is for the Collaboration to verify that all points raised in the first comment period have been addressed, though on occasion a significant new issue may still be raised at this step

Route to publication

- Upon submission to arXiv the paper is made publicly available on the CERN Document Server and on the ALICE web site.
- The response from the journal referee(s) is made available to the Collaboration via the corresponding website.
- The PC prepares a revised manuscript and a response to the referees' comments.
- The IRC reviews the modified manuscript and response to the referees' comments, and recommends corrections and changes as necessary.
- If the paper is rejected by the journal or changes requested by the journal are deemed unacceptable to the Collaboration, appeal or resubmission to a different journal will be considered