ATLAS SimpleAnalysis:
You can handle the truth!

Jeanette Lorenz (LMU Munich)

On behalf of the ATLAS Collaboration

(Re)interpreting the results of new physics searches at the LHC / 17.02.21

Why we need truth-level analyses

Detector-level simulation is not always available or feasible to be used

→ A truth-level analysis allows to implement an analysis also without the knowledge of the detector.

Used both inside and outside of experimental collaborations:

- *External:*

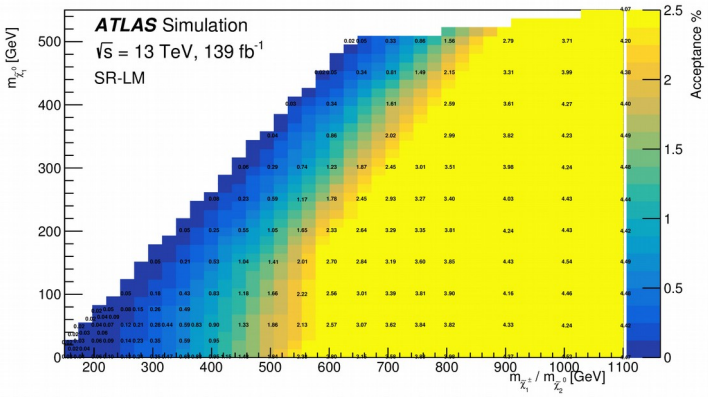
  → Reinterpretation of analyses & suggesting new analyses.

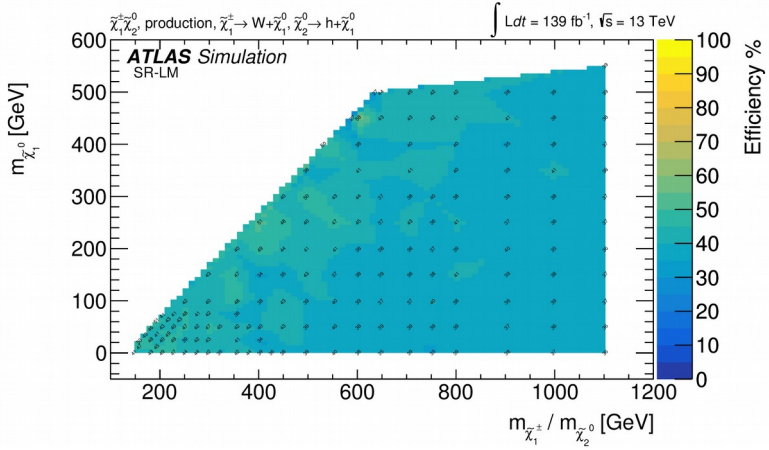
- *Internal:*

  → Calculation of acceptances and efficiencies.

  → Calculation of some theoretical uncertainties, if full simulation too expensive.

  → In reinterpretation efforts like scans of the phenomenological MSSM.



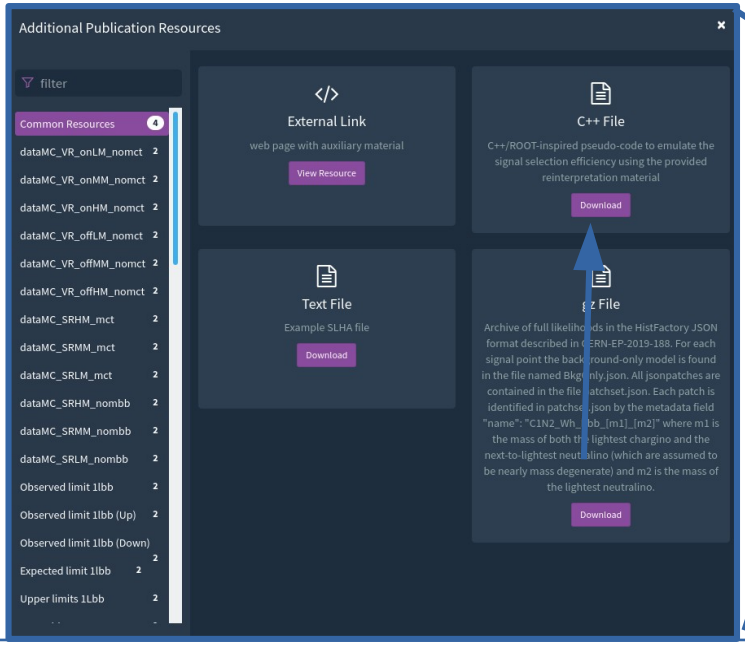


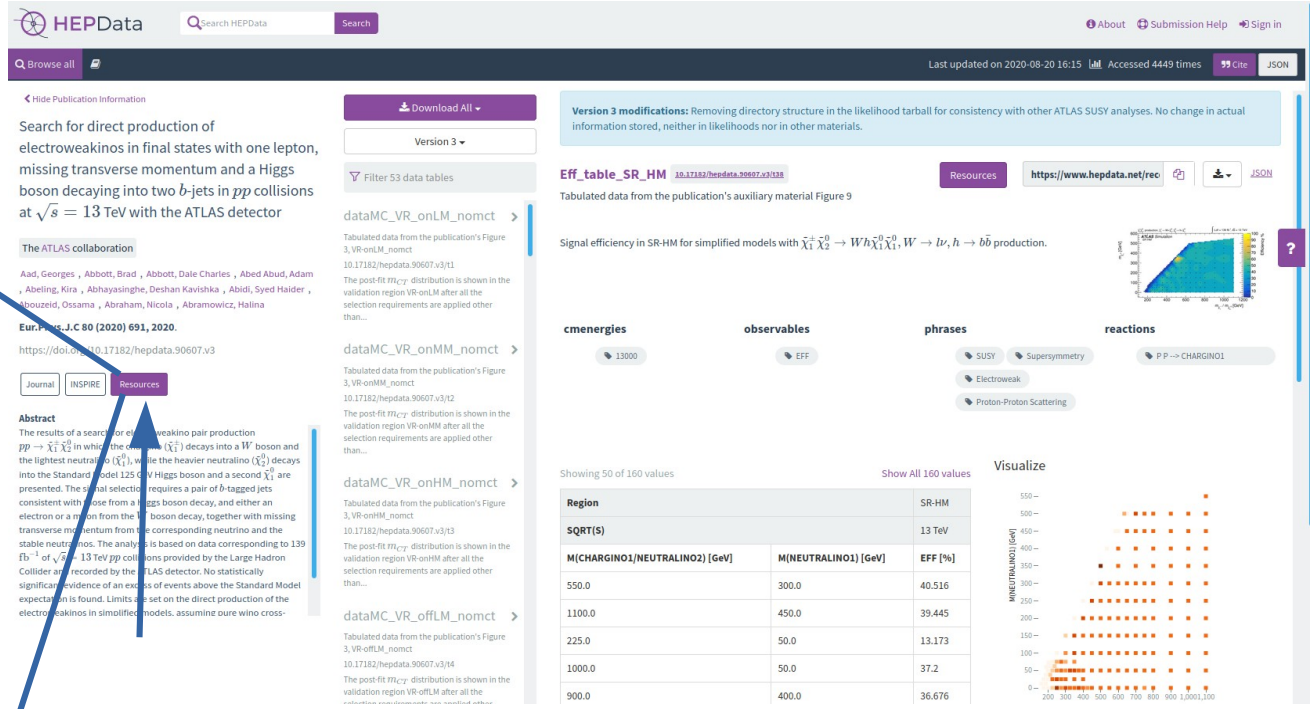
[https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/SUSY-2019-08/]

Truth code snippets available in HEPData

[https://www.hepdata.net/record/ins1755298]

ATLAS SUSY searches release a C++ pseudocode for each analysis within HEPData since a while.





Pseudocode

File Edit View Projects Bookmarks Sessions Tools Settings Help

Documents

EwkOneLeptonTwoBjets2018.cxx

```
#include "SimpleAnalysis/AnalysisClass.h"
#include "SimpleAnalysis/NtupleMaker.h"
#include "SimpleAnalysis/PDFReweight.h"
#include <LHAPDF/LHAPDF.h>
#include "TMath.h"

DefineAnalysis(EwkOneLeptonTwoBjets2018)
// Wh->1l+bb+met analysis (Run2 data)

void EwkOneLeptonTwoBjets2018::Init()

{
  // Define signal/control regions
  addRegions({"SR_h_Low", "SR_h_Med", "SR_h_High"} );
  addRegions({"SR_h_Low_Incl", "SR_h_Med_Incl", "SR_h_High_Incl"} );
  addRegions({"SR_h_Low_bin1", "SR_h_Low_bin2", "SR_h_Low_bin3"});
  addRegions({"SR_h_Med_bin1", "SR_h_Med_bin2", "SR_h_Med_bin3"});
  addRegions({"SR_h_High_bin1", "SR_h_High_bin2", "SR_h_High_bin3"});
  addRegions({"CR_tt_Low", "CR_tt_Med", "CR_tt_High"});
  addRegions({"WCR", "STCR"});
  addRegions({"VR_off_Low", "VR_off_Med", "VR_off_High"});
  addRegions({"VR_on_Low", "VR_on_Med", "VR_on_High"});
}

void EwkOneLeptonTwoBjets2018::ProcessEvent(AnalysisEvent *event)
{
  //
  // Soft Objects
  //
  auto  Soft_Elec = event->getElectrons(7,2.47,ELooseBLLH | EZ05mm);
  auto  Soft_Muon = event->getMuons(6,2.70, MuMedium| MuNotCosmic | MuGood | MuZ05mm);
  auto  Soft_Jets = event->getJets(20.,4.5);
  auto  met_Vect  = event->getMET();
  float met       = met_Vect.Pt();
  //
  // Baseline objects
  //
  // Reject events with bad jets
  auto radiusCalcLep = [] (const AnalysisObject& lep,const AnalysisObject&) {return (0.04 + 10/lep.Pt()) > 0.4 ? 0.4 : 0.04 + 10/lep.Pt();};
  auto Base_Elec     = overlapRemoval(Soft_Elec,Soft_Elec,0.01);
  Base_Elec      = overlapRemoval(Soft_Elec,Soft_Muon,0.01);
//  auto Base_Jets      = overlapRemoval(Soft_Jets,Base_Elec,0.2, NOT(BTag77MV2c10));
//  Base_Elec           = overlapRemoval(Base_Elec,Base_Jets,radiusCalcLep);
  Base_Elec           = overlapRemoval(Base_Elec,Soft_Jets,radiusCalcLep);
  auto Base_Jets      = overlapRemoval(Soft_Jets, Soft_Muon, 0.2);
  auto Base_Muon      = overlapRemoval(Soft_Muon,Base_Jets,radiusCalcLep);

   auto Signal_Elec = filterObjects(Base_Elec,7.,2.47,ETightLH | ED0Sigma5  | EZ05mm  | EIsoGradientLoose);
  auto Signal_Muon = filterObjects(Base_Muon,6.,2.7,MuMedium | MuD0Sigma3 | MuZ05mm |MuIsoGradientLoose);
  auto Signal_Jets = filterObjects(Base_Jets,30.,2.80,JVT50Jet);
  auto Signal_BJet = filterObjects(Signal_Jets,30.,2.8,BTag77MV2c10);

  //
  auto Signal_Lep  = Signal_Elec + Signal_Muon;
  //Count the number of signal objects
  unsigned int N_Signal_Elec = Signal_Elec.size();
  unsigned int N_Signal_Muon = Signal_Muon.size();
  unsigned int N_Signal_Lept = N_Signal_Elec + N_Signal_Muon;
  unsigned int N_Signal_Jets = Signal_Jets.size();
  unsigned int N_Signal_BJet = Signal_BJet.size();
  float mt=0, m_CT=0,  mbb=0, b2jetpt=0, mlb1=0;
  if (Signal_Lep.size()==1   && Signal_BJet.size()==2 ) {
    mt   = calcMT(Signal_Lep[0], met_Vect);
    m_CT = calcMCT(Signal_BJet[0],Signal_BJet[1]);
    mbb= (Signal_BJet[0]+Signal_BJet[1]).M();
    mlb1= (Signal_BJet[0]+Signal_Lep[0]).M();
        b2jetpt = Signal_BJet[1].Pt();
  }

  //Define ntupleVar variables for Hamish
  int SR_h_Low(0), SR_h_Med(0), SR_h_High(0);
  int SR_h_Low_Incl(0), SR_h_Med_Incl(0), SR_h_High_Incl(0);
```

- Definition and selection of objects
- Precise cut sequence to define signal regions (and possibly other regions).

```
// Preselection
if(N_Signal_Lept != 1) return;
if(N_Signal_BJet != 2) return;
if(N_Signal_Jets>3 || N_Signal_Jets < 2) return;
if(mt< 50.) return;

 if(met< 220.) return;
// Signal regions
if(met>240 && mbb>100 && mbb<=140 && m_CT>180){
  if(mt>100 && mt<160){
            accept("SR_h_Low");
            SR_h_Low = 1;
            if(m_CT > 180 && m_CT <= 230){
                    accept("SR_h_Low_bin1");
                    SR_h_Low_bin1 = 1;
            }else if(m_CT > 230 && m_CT <= 280){
                    accept("SR_h_Low_bin2");
                    SR_h_Low_bin2 = 1;
            }else{
                    accept("SR_h_Low_bin3");
                    SR_h_Low_bin3 = 1;
            }
        }
  else if(mt>160 && mt<240){
            accept("SR_h_Med");
            SR_h_Med = 1;
            if(m_CT > 180 && m_CT <= 230){
                    accept("SR_h_Med_bin1");
                    SR_h_Med_bin1 = 1;
            }else if(m_CT > 230 && m_CT <= 280){
                    accept("SR_h_Med_bin2");
                    SR_h_Med_bin2 = 1;
            }else{
                    accept("SR_h_Med_bin3");
                    SR_h_Med_bin3 = 1;
            }
        }
  else if(mt>240 && mlb1 > 120){
            accept("SR_h_High");
            SR_h_High = 1;
            if(m_CT > 180 && m_CT <= 230){
                    accept("SR_h_High_bin1");
                    SR_h_High_bin1 = 1;
            }else if(m_CT > 230 && m_CT <= 280){
                    accept("SR_h_High_bin2");
                    SR_h_High_bin2 = 1;
            }else{
                    accept("SR_h_High_bin3");
                    SR_h_High_bin3 = 1;
```

Zooming in on object definitions

p_T of object

η of object

Working point

Definition of basic objects: electrons, muons, jets,…

```
//
// Soft Objects
//
auto baseEle = event->getElectrons(7,2.47, ELooseBLLH);
auto baseMuon = event->getMuons(6,2.70, MuMedium | MuNotCosmic | MuZ05mm | MuQoPSignificance);
auto baseJets = event->getJets(20.,4.5);
auto met_Vect  = event->getMET();
float met       = met_Vect.Pt();
auto weights        = event->getMCWeights();
```

Precise sequence of overlap removal between baseline objects

```
//
// Baseline objects
//
auto radiusCalcLep = [] (const AnalysisObject& lep,const AnalysisObject&) {
                                                return (0.04 + 10/lep.Pt()) > 0.4 ? 0.4 : 0.04 + 10/lep.Pt();
                                        };

// ele-mu OR
baseEle = overlapRemoval(baseEle, baseMuon, 0.01);
// ele-jet OR
baseJets = overlapRemoval(baseJets, baseEle, 0.2);
baseEle = overlapRemoval(baseEle, baseJets, radiusCalcLep);
// mu-jet OR
baseJets = overlapRemoval(baseJets, baseMuon, 0.2, LessThan3Tracks);
baseMuon = overlapRemoval(baseMuon, baseJets, radiusCalcLep);
```

Zooming in on object definitions

Second stage of object selection – tighter quality criteria, used for the search regions

```
    // Get signal electrons with FCLoose for pT < 200 GeV and FCHighPtCaloOnly for pT > 200
    // filterObjects() only allows lower pT cut, so work around that
    auto signalEle = filterObjects(baseEle,7., 2.47, ETightLH | ED0Sigma5  | EZ05mm  | EIsoFCLoose);
    AnalysisObjects signalEleLowPt;
for (const auto& ele : signalEle) {
  if ((ele.Pt() < 200.)) signalEleLowPt.push_back(ele);
    }
    auto signalEleHighPt = filterObjects(baseEle,200., 2.47, ETightLH | ED0Sigma5  | EZ05mm  | EIsoFCHighPtCaloOnly);
    signalEle = signalEleLowPt + signalEleHighPt;

    auto signalMuon = filterObjects(baseMuon,6., 2.7, MuD0Sigma3 | MuZ05mm | MuIsoFCLoose);
    auto signalLept =  signalEle + signalMuon;

    auto signalJets = filterObjects(baseJets, 30., 2.80, JVT120Jet);
    auto signalBJets = filterObjects(signalJets, 30., 2.8, BTag77MV2c10);
```

Zooming in on selection criteria

Preselection

```
if(mbb>100 && mbb<=140) accept("preset_mbb");

// Preselection
if(N_baseLept != 1) return;
if(N_signalLept != 1) return;
if(N_signalJets>3 || N_signalJets < 2) return;
if(N_signalBJets != 2) return;
if(mt< 50.) return;
if(met< 220.) return;
```

Search regions (signal, control, validation regions)

```
// Signal regions
if(met>240 && mbb>100 && mbb<=140 && m_CT>180) {
        if(mt>100 && mt<160) {
                accept("SR_h_Low");
                if(m_CT > 180 && m_CT <= 230) {
                        accept("SR_h_Low_bin1");
                }else if(m_CT > 230 && m_CT <= 280) {
                        accept("SR_h_Low_bin2");
                }else{
                        accept("SR_h_Low_bin3");
                }
        }
        else if(mt>160 && mt<240) {
                accept("SR_h_Med");
                if(m_CT > 180 && m_CT <= 230) {
                        accept("SR_h_Med_bin1");
```

accept(region) indicates that we fill in this event for the named region

Pseudocode

```
//
// Reject events with bad jets
auto radiusCalcLep = [] (const AnalysisObject& lep,const AnalysisObject&) {return (0.04 + 10/lep.Pt()) > 0.4 ? 0.4 : 0.04 + 10/lep.Pt();};
auto Base_Elec      = overlapRemoval(Soft_Elec,Soft_Elec,0.01);
Base_Elec      = overlapRemoval(Soft_Elec,Soft_Muon,0.01);
//  auto Base_Jets      = overlapRemoval(Soft_Jets,Base_Elec,0.2, NOT(BTag77MV2c10));
//  Base_Elec           = overlapRemoval(Base_Elec,Base_Jets,radiusCalcLep);
Base_Elec           = overlapRemoval(Base_Elec,Soft_Jets,radiusCalcLep);
auto Base_Jets      = overlapRemoval(Soft_Jets, Soft_Muon, 0.2);
auto Base_Muon      = overlapRemoval(Soft_Muon,Base_Jets,radiusCalcLep);

 auto Signal_Elec = filterObjects(Base_Elec,7.,2.47,ETightLH | ED0Sigma5  | EZ05mm  | EIsoGradientLoose);
auto Signal_Muon = filterObjects(Base_Muon,6.,2.7,MuMedium | MuD0Sigma3 | MuZ05mm |MuIsoGradientLoose);
auto Signal_Jets = filterObjects(Base_Jets,30.,2.80,JVT50Jet);
auto Signal_BJet = filterObjects(Signal_Jets,30.,2.8,BTag77MV2c10);

//
auto Signal_Lep=  Signal_Elec + Signal_Muon;
//Count the number of signal objects
unsigned int N_Signal_Elec = Signal_Elec.size();
unsigned int N_Signal_Muon = Signal_Muon.size();
unsigned int N_Signal_Lept = N_Signal_Elec + N_Signal_Muon;
unsigned int N_Signal_Jets = Signal_Jets.size();
unsigned int N_Signal_BJet = Signal_BJet.size();
float mt=0, m_CT=0,  mbb=0, b2jetpt=0, mlb1=0;
if (Signal_Lep.size()==1   && Signal_BJet.size()==2 ) {
  mt   = calcMT(Signal_Lep[0], met_Vect);
  m_CT = calcMCT(Signal_BJet[0],Signal_BJet[1]);
  mbb= (Signal_BJet[0]+Signal_BJet[1]).M();
  mlb1= (Signal_BJet[0]+Signal_Lep[0]).M();
      b2jetpt = Signal_BJet[1].Pt();
}

//Define ntupleVar variables for Hamish
int SR_h_Low(0), SR_h_Med(0), SR_h_High(0);
int SR_h_Low_Incl(0), SR_h_Med_Incl(0), SR_h_High_Incl(0);
```

So far helper functions not public
→ Release of SimpleAnalysis!

SimpleAnalysis

We are now releasing the code base that is used for truth-level SUSY studies within ATLAS.
...in multiple stages

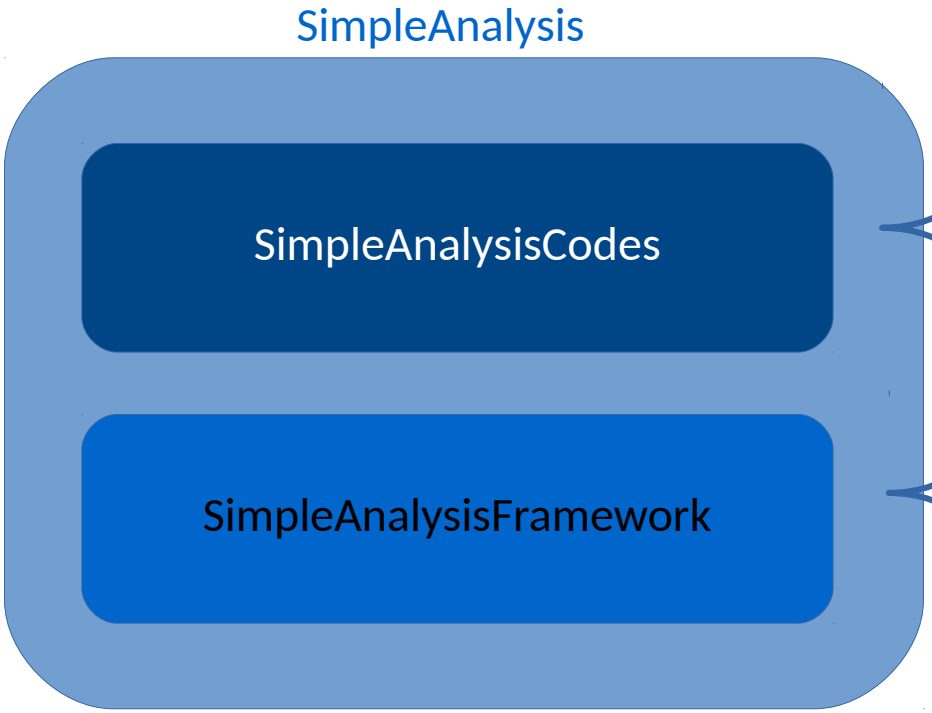
Benefits:

- The full code is available, including the helper functions – not only the analysis implementation.
  → Visible how the different pieces connect.
- The code can actually be run.
- We provide a central location for all analyses implemented in SimpleAnalysis
  (~most of the Run 2 SUSY program).

Caveats:

- So far only works with the ATLAS-internal truth format, or with flat ROOT n-tuples containing 4-vectors of leptons, jets, ...
- ATLAS-internal also smearing functions are used to correct the truth-level to the detector response.
  → We do not release these functions.

Structure




```
├── LICENSE
├── README.md
├── SimpleAnalysisCodes
│   ├── CMakeLists.txt
│   ├── data
│   │   ├── StopOneLepton2016_BDT-tN_diag_high.weights1.xml
│   │   ├── StopOneLepton2016_BDT-tN_diag_high.weights2.xml
│   │   ├── StopOneLepton2016_BDT-tN_diag_low.weights1.xml
│   │   ├── StopOneLepton2016_BDT-tN_diag_low.weights2.xml
│   │   ├── StopOneLepton2016_BDT-tN_diag_med.weights1.xml
│   │   └── StopOneLepton2016_BDT-tN_diag_med.weights2.xml
│   └── src
│       └── ANA-SUSY-2016-16.cxx
├── SimpleAnalysisFramework
│   ├── CMakeLists.txt
│   ├── LICENSE
│   ├── README.md
│   ├── SimpleAnalysisFramework [14 entries exceeds filelimit, not opening dir]
│   ├── src [33 entries exceeds filelimit, not opening dir]
│   └── util
│       ├── execBase.h
│       ├── simpleAnalysis.cxx
│       └── slimMaker.cxx
└── docs
    ├── mkdocs.yml
    ├── requirements.txt
    └── src
```

Analysis codes and required data files (MVAs, etc.)

SimpleAnalysis core code

Link!

How does it work?

Because of the dependency of the ATLAS-internal file format at the moment (although flat n-tuples can be used instead), use of a docker image is necessary:

```
docker pull gitlab-registry.cern.ch/atlas-sa/simple-analysis
docker run -it --rm gitlab-registry.cern.ch/atlas-sa/simple-analysis

(docker) source /release_setup.sh
```

Running an analysis is simply done by naming the analysis and the input file:

```
(docker) simpleAnalysis -a <analysisOfChoice> <inputFile>
```

A tutorial is available at https://simpleanalysis.docs.cern.ch/

How to find an analysis

https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/SUSY-2016-06/

Meistbesucht Deutscher Alpenverei...

AtlasPublic Web Physics Groups Other Groups

TWiki.org

Analysis identifier

Search for long-lived charginos based on a disappearing-track signature in pp collisions at $\sqrt{s}$ = 13 detector

JHEP 06 (2018) 022

6 December 2017

Contact: ATLAS SUSY conveners

Content
e-print arXiv:1712.02118
Inspire record
Data points
Figures Tables Auxiliary Material

- ANA-SUSY-2016-06.cxx
- ANA-SUSY-2016-09.cxx
- ANA-SUSY-2016-10.cxx
- ANA-SUSY-2016-11.cxx

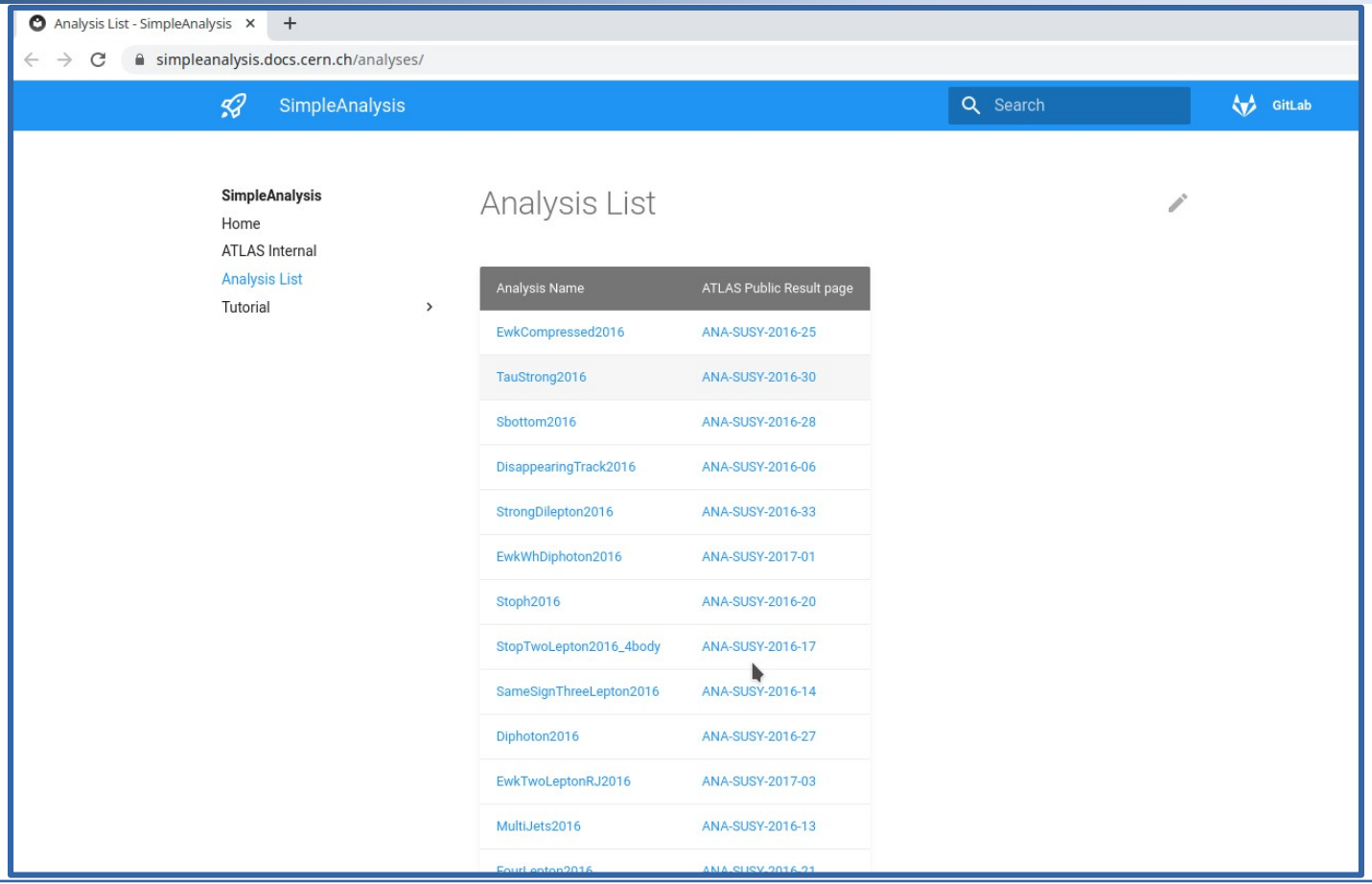
Analysis name within SimpleAnalysis file

ANA-SUSY-2016-06.cxx 6.73 KB

```
#include "SimpleAnalysisFramework/AnalysisClass.h"
#include <TRandom.h>
#include <TFile.h>
#include <TH2.h>
#include <TF1.h>

DefineAnalysis(DisappearingTrack2016)
```

List of available analyses



Webpage available to link analysis identifiers to analysis names: Link

Summary and plans

- Code snippets for ATLAS SUSY searches available in HEPData since a while.
- Now we also release the framework behind this code, to give the context and explain the helper functions
  → Analysis workflow now fully transparent.

- On the longer term, we plan to allow for additional file formats as input to SimpleAnalysis.
- This is also the first time ATLAS releases code to be actually used by others, so we are happy to hear feedback.

Please address comments and questions to:
atlas-phys-susy-conveners@cern.ch