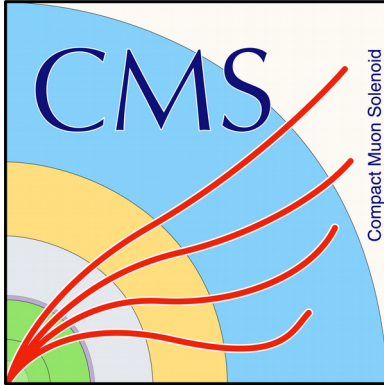


Introduction to analysis with PAF



A single group, a single framework,
one code, hundred analysis.

PAF – Proof Analysis Framework

PAF is tool that easily allows to use different PROOF environments in your analysis. Also gives the base to build you analysis looping over the entries of a single tree.

Main utilities:

- Pass sample-dependent parameters to the analyzer.
- Loop over all the entries using a PROOF environment
- Use different chained selectors.
- Produce a single output with trees and histograms created in the analysis.



Further information, all the documentation and tutorials:

<http://www.hep.uniovi.es/PAF/>

This framework is based in PAF selectors, prepared to run on HepPy Trees and using Dataset Manager to obtain the input trees.

Dataset Manager

Dataset Manager is a tool to collect all the info related to the available samples, typically using a shared spreadsheet.

In this analysis the samples will be listed here:

<https://docs.google.com/spreadsheets/d/1b4qnWfZrimEGYc1z4dHl21-A9qyJgpqNUbhOlvCzjbE/>

Where each layer corresponds to a different production of HeppyTrees. The sheet contains info about the path of the trees (O1) name of the samples, name of the datasets, number of events, cross section, etc.

More info about usage of the Dataset Manager here:

<https://gitlab.cern.ch/IFCA-UO-CMS/Utils/tree/master/DatasetManager>

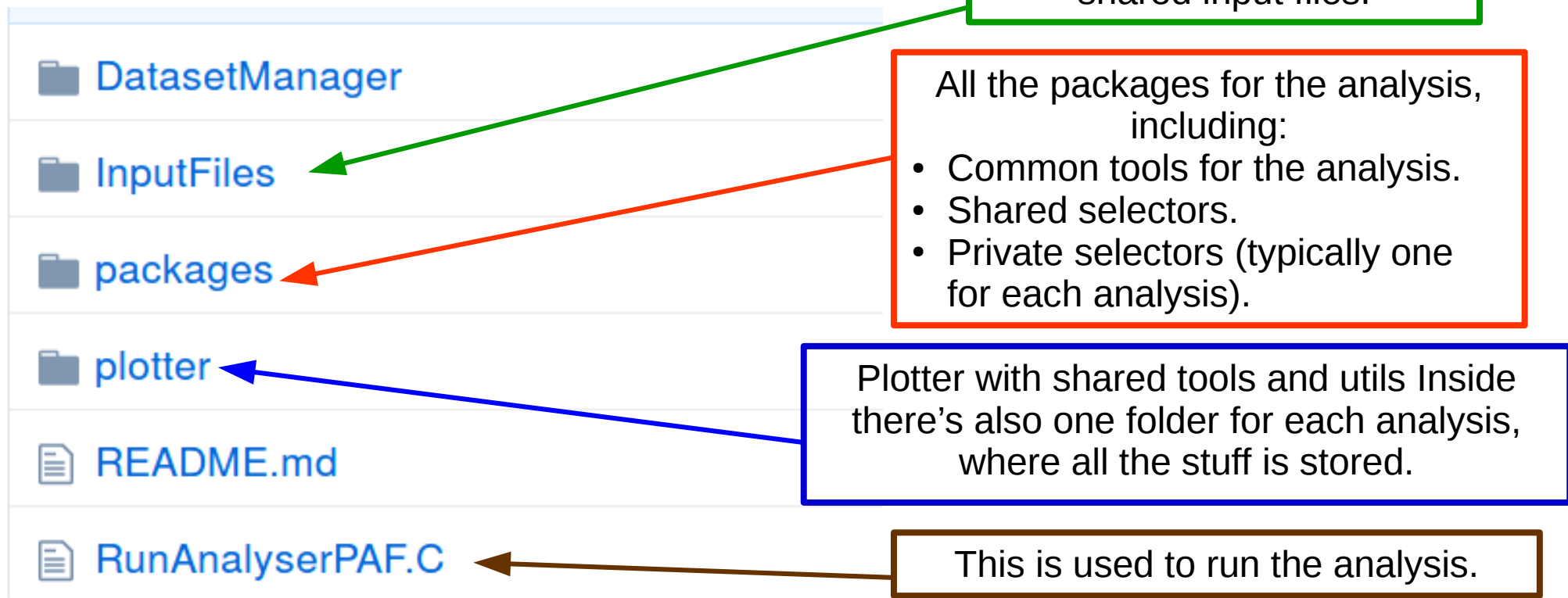
The repository

The analyzer will be located in a git repository:

[<https://github.com/Oviedo-PAF/AnalysisPAF>]

The different analysis share most of the code but also have their own directories where users can freely create and modify all the files.

In the top level we can find:



The packages

 BTagSFUtil CreateMiniTree EventBuilder Functions Jet JetSelector Lepton LeptonSF LeptonSelector PUWeight StopAnalysis TopAnalysis WWAnalysis mt2

Packages in **RED** (BtagSFUtil, Functions, LeptonSF, PUWeight). Shared packages. In principle you don't have to touch them. Only if you need to add some new SF, update weights, etc.

Packages in **BROWN** (Jet, Lepton). You don't want to change anything here.

Packages in **BLUE** (LeptonSelector, JetSelector, EventBuilder) – Shared selectors! Use them carefully to select your objects/trigger/etc. Don't modify the shared code!

Packages in **GREEN** (StopAnalysis, TopAnalysis, WWAnalysis) – One (or more!) for each analysis... use only the ones you own.

Packages in **YELLOW** (mt2) – Extra packages... you can add extra **useful** packages for everyone!

Jets and Leptons

In this framework objects “Jet” and objects “Lepton” are used.
Basically: TLorentzVectors with some other properties...

Leptons have: charge, type (0 = Muon, 1 = Elec), bool isElec, bool isMuon. Leptons contain SF:

Float_T GetSF(**Int_t** s) → s = 0 (nominal), 1 (up variation), -1 (down variation).

Jets have: mcFlavour, csv. IsBtag.

For systematic: pTJESUp, pTJESDown, ptJERUp...

isBtag_BtagUp, isBtag_BtagDown, isBtag_MisTagUp...

Probably, **taus** coming soon... (used in ttH analysis).

Package Funtions

Enum definitions used in all th analysis are here.

```
13  enum iSFs{
14      iMuonReco, iMuonId, iMuonIdSUSY, iMuonIsoTightId, iMuonIsoMediumId, iMuonIsoSUSY, iMuonIP2D, iMuonSIP3D,
15      iElecReco, iElecId, iElecIdSUSY, iElecIso, iElecIsoSUSY, iElecIP2D, iElecSIP3D,
16      iTrigDoubleMuon, iTrigDoubleElec, iTrigElMu,
17      nfiles
18  };
19
20  enum wps{iVeto, iLoose, iMedium, iTight, iVeryTight, iExtreamlyTight, iWPforStop};
21
22  enum sel{iStopSelec, iTopSelec, iTWSelec, iWWSelec, ittDMSelec, ittHSelec};
23
24  enum sys{iNom,
25      iJesUp, iJesDown, iJERUp, iJERDown,
26      iBtagUp, iBtagDown, iBtagMisUp, iBtagMisDown,
27      iLepEffUp, iLepEffDown,
28      iTrigUp, iTrigDown,
29      iPUUp, iPUDown,
30      iFSUp, iFSDown,
31      nSys
32  };
```

If you need more
working points, just
add them here!

Apart from that, shared useful general functions are defined here.
Don't hesitate to use them and share your general useful
functions here!!!

BtagSFUtil and PUWeights

This packages are updated with newest **PU weights** and **bTag SFs**. Several PU profiles are available.

Typically, **you won't need to code nothing in this packages**, just use them in your analysis to get the SFs.

```
if (OperatingPoint=="Loose") {
    TaggerOP += "L";
    if (TaggerName=="CSV") TaggerCut = 0.244;
    //if (TaggerName=="CSVv2") TaggerCut = 0.605;
    if (TaggerName=="CSVv2") TaggerCut = 0.5426; // for Moriond17
    reader_bc = new BTagCalibrationReader(&calib, BTagEntry::OP_LOOSE, MeasurementType, SystematicFlagL);
    //reader_1 = new BTagCalibrationReader(&calib, BTagEntry::OP_LOOSE, MeasurementType, SystematicFlagL);
    //reader_1 = new BTagCalibrationReader(&calib, BTagEntry::OP_LOOSE, "comb", SystematicFlagL);
    reader_1 = new BTagCalibrationReader(&calib, BTagEntry::OP_LOOSE, "incl", SystematicFlagL);
} else if (OperatingPoint=="Medium") {
    TaggerOP += "M";
    if (TaggerName=="CSV") TaggerCut = 0.679;
    //if (TaggerName=="CSVv2") TaggerCut = 0.890; // for 74X
    //if (TaggerName=="CSVv2") TaggerCut = 0.800; // for 76X
    if (TaggerName=="CSVv2") TaggerCut = 0.8484; // for Moriond17
    reader_bc = new BTagCalibrationReader(&calib, BTagEntry::OP_MEDIUM, MeasurementType, SystematicFlagL);
    //reader_1 = new BTagCalibrationReader(&calib, BTagEntry::OP_MEDIUM, MeasurementType, SystematicFlagL);
    //reader_1 = new BTagCalibrationReader(&calib, BTagEntry::OP_MEDIUM, "comb", SystematicFlagL);
    reader_1 = new BTagCalibrationReader(&calib, BTagEntry::OP_MEDIUM, "incl", SystematicFlagL);
} else if (OperatingPoint=="Tight") {
    TaggerOP += "T";
    if (TaggerName=="CSV") TaggerCut = 0.898;
    if (TaggerName=="TCHP") TaggerCut = 3.41;
```

*Missing MC btag
efficiencies for loose
b-tagging working point*

WPs for Moriond 17

Package LeptonSF

SFs and WPs defined here! Check that yours are here and the root files are in the folder.

```
void LeptonSF::loadHisto(Int_t iHisto, Int_t wp){  
    TString filename = ""; TString histoname = "  
// >>>>>>>>>>>>>>>>>>>>> Muons  
    if(iHisto == iMuonReco){  
        filename = "Tracking_EfficienciesAndSF_BCDEFGH"; histoname = "ratio_eff_eta3_dr030e030_corr";  
        fMuonTrackerSF = LoadTrackerMuonsSF(path_to_SF_histos + filename + ".root", histoname);  
    }  
    else if(iHisto == iMuonIdSUSY){  
        if      (wp == iLoose){ filename = "SUS_MuonLooseIdM17"; histoname = "SF";}   
        else if(wp == iMedium){ filename = "SUS_MuonMediumIdM17"; histoname = "SF";}   
        else if(wp == iTight){ filename = ""; histoname = "";}   
        fMuonIdSFSUSY = GetHistogramFromFileD(path_to_SF_histos + filename + ".root", histoname, "fMuonIdSFSUSY");  
    }  
    else if(iHisto == iMuonId){  
        if(wp == iLoose){  
            filename = "MuonSFId_BCDEF"; histoname = "MC_NUM_LooseID_DEN_genTracks_PAR_pt_eta/pt_abseta_ratio";  
            fMuonIdSF_BCDEF = GetHistogramFromFileD(path_to_SF_histos + filename + ".root", histoname, "fMuonIdSF_BCDEF");  
            filename = "MuonSFId_GH"; histoname = "MC_NUM_LooseID_DEN_genTracks_PAR_pt_eta/pt_abseta_ratio";  
            fMuonIdSF_GH     = GetHistogramFromFileD(path_to_SF_histos + filename + ".root", histoname, "fMuonIdSF_GH");  
        }  
  
        if(wp == iMedium){  
            filename = "MuonSFId_BCDEF"; histoname = "MC_NUM_MediumID_DEN_genTracks_PAR_pt_eta/pt_abseta_ratio";  
            fMuonIdSF_BCDEF = GetHistogramFromFileD(path_to_SF_histos + filename + ".root", histoname, "fMuonIdSF_BCDEF");  
            filename = "MuonSFId_GH"; histoname = "MC_NUM_MediumID_DEN_genTracks_PAR_pt_eta/pt_abseta_ratio";  
            fMuonIdSF_GH     = GetHistogramFromFileD(path_to_SF_histos + filename + ".root", histoname, "fMuonIdSF_GH");
```

Package LeptonSF

In the lepton selector, the SF is assigned to each lepton with these functions.

```
Float_t LeptonSF::GetLeptonSF(Float_t pt, Float_t ieta, Int_t type){  
    Float_t eta = TMath::Abs(ieta);  
    if(pt > 120) pt = 119;  
    Int_t nSFs = loadedHistos.size();  
    Float_t SF = 1; Int_t id;  
    for(Int_t i = 0; i < nSFs; i++){  
        id = loadedHistos[i];  
        if(type == 0){  
            if (id == iMuonReco)    SF *= GetTrackerMuonSF(eta);  
            else if(id == iMuonIdSUSY) SF *= fMuonIdSFSUSY      ->GetBinContent(fMuonIdSFSUSY      ->FindBin(pt, eta));  
            else if(id == iMuonId)    SF *= (fMuonIdSF_BCDEF->GetBinContent(fMuonIdSF_BCDEF->FindBin(pt, eta))*lumi  
            else if(id == iMuonIsoSUSY) SF *= fMuonIsoSFSUSY      ->GetBinContent(fMuonIsoSFSUSY      ->FindBin(pt, eta));  
            else if(id == iMuonIsoMediumId || id == iMuonIsoTightId)  
                SF *= (fMuonIsoSF_BCDEF->GetBinContent(fMuonIsoSF_BCDEF->FindBin(pt, eta))*lumiBCDEF + fMuonIsoSF  
            else if(id == iMuonIP2D)    SF *= fMuonIP2DSF      ->GetBinContent(fMuonIP2DSF      ->FindBin(pt, eta));  
            else if(id == iMuonSIP3D)    SF *= fMuonSIP3DSF      ->GetBinContent(fMuonSIP3DSF      ->FindBin(pt, eta));  
        }  
        else if(type == 1){  
            if(id == iElecReco)    SF *= fElecTrackerSF ->GetBinContent(fElecTrackerSF->FindBin(eta, 50));  
            else if(id == iElecIdSUSY)    SF *= fElecIdSF      ->GetBinContent(fElecIdSF      ->FindBin(pt, eta));  
            else if(id == iElecIsoSUSY) SF *= fElecIsoSF      ->GetBinContent(fElecIsoSF      ->FindBin(pt, eta));  
            else if(id == iElecId)    SF *= fElecIdSF      ->GetBinContent(fElecIdSF      ->FindBin(eta, pt));  
            else if(id == iElecIP2D)    SF *= fElecIP2DSF      ->GetBinContent(fElecIP2DSF      ->FindBin(eta, pt));  
            else if(id == iElecSIP3D)    SF *= fElecSIP3DSF      ->GetBinContent(fElecSIP3DSF      ->FindBin(eta, pt));  
        }  
    }  
}
```

Package LeptonSF

Scale factors for trigger: **not available yet!!**

They'll be implemented here, and used as lepton SFs.

```
Float_t LeptonSF::GetTrigDoubleMuSF(Float_t eta1, Float_t eta2) const { // binned in eta1, eta2
    eta1 = TMath::Abs(eta1);
    eta2 = TMath::Abs(eta2);
    return fDoubleMuSF->GetBinContent(fDoubleMuSF->FindBin(eta1, eta2));
}

Float_t LeptonSF::GetTrigDoubleElSF(Float_t eta1, Float_t eta2) const { // binned in eta1, eta2
    eta1 = TMath::Abs(eta1);
    eta2 = TMath::Abs(eta2);
    return fDoubleElSF->GetBinContent(fDoubleElSF->FindBin(eta1, eta2));
}

Float_t LeptonSF::GetTrigElMuGSF(Float_t eta1, Float_t eta2) const { // binned in eta1, eta2
    eta1 = TMath::Abs(eta1);
    eta2 = TMath::Abs(eta2);
    return fMuEGSF->GetBinContent(fMuEGSF->FindBin(eta1, eta2) );
}

// Trigger SF errors
Float_t LeptonSF::GetTrigDoubleMuSF_err(Float_t eta1, Float_t eta2) const { // binned in eta1, eta2
    return fDoubleMuSF->GetBinError(fDoubleMuSF->FindBin(TMath::Abs(eta1), TMath::Abs(eta2)));
}

Float_t LeptonSF::GetTrigDoubleElSF_err(Float_t eta1, Float_t eta2) const { // binned in eta1, eta2
    return fDoubleElSF->GetBinError(fDoubleElSF->FindBin(TMath::Abs(eta1), TMath::Abs(eta2)));
}
```

Lepton Selector

Select your SFs depending on your working points...

[illegible]

Lepton Selector

```
//#####  
//## Definition of wps...  
//#####  
Bool_t LeptonSelector::getSIPcut(Float_t cut){  
    if(sip > cut) return false;  
    return true;  
}  
  
Bool_t LeptonSelector::getGoodVertex(Int_t wp){  
    if(wp == iMedium && (dxy > 0.2 || dz > 0.5)) return false;  
    if(wp == iTight && (dxy > 0.05 || dz > 0.1)) return false;  
    return true;  
}  
  
Bool_t LeptonSelector::getRelIso03P0G(Int_t wp){  
    if(type == 1){ // electrons  
        if(wp == iWPforStop && RelIso03 > 0.12) return false;  
        if(etaSC <= 1.479){  
            if(wp == iVeto && RelIso03 > 0.1750) return false;  
            if(wp == iLoose && RelIso03 > 0.0994) return false;  
            if(wp == iMedium && RelIso03 > 0.0695) return false;  
            if(wp == iTight && RelIso03 > 0.0588) return false;  
        }  
        else if(etaSC > 1.479){  
            if(wp == iVeto && RelIso03 > 0.1590) return false;  
            if(wp == iLoose && RelIso03 > 0.1070) return false;  
            if(wp == iMedium && RelIso03 > 0.0821) return false;  
            if(wp == iTight && RelIso03 > 0.0571) return false;  
        }  
    }  
}
```

Check that the
WPs you're going
to use are defined!

If not, add them to
the list.

Lepton Selector

Here define the selected leptons for your analysis!

```
#####  
## Lepton definitions for each analysis  
#####  
  
//  
// Use the functions above to define your objects  
//  
//===== SELECTED LEPTONS  
Bool_t LeptonSelector::isGoodLepton(Lepton lep){  
    Bool_t passId; Bool_t passIso;  
    if(gSelection == iStopSelec){  
        // Tight cut-based electrons,  $p_T > 20$ ,  $|\eta| < 2.4$ , RelIso POG, tightIP2D, SIP3D > 4  
        // Medium Muon ID, RelIso POG, tightIP2D, SIP3D > 4  
        if(lep.isMuon){  
            passId = getMuonId(iMedium);  
            passIso = getRelIso03POG(iWPforStop);  
        }  
        if(lep.isElec){  
            passId = getElecCutBasedId(iWPforStop);  
            passIso = getRelIso03POG(iWPforStop);  
            if(TMath::Abs(etaSC) > 1.4442 && TMath::Abs(etaSC) < 1.566) return false;  
        }  
        if(lep.p.Pt() < 20 || TMath::Abs(lep.p.Eta()) > 2.4) return false;  
        if(passId && passIso && getGoodVertex(iTight) && getSIPcut(4)) return true;  
        else return false;  
    }  
    else if(gSelection == iTopSelec || gSelection == iTWSelec){  
        // Tight cut-based electrons,  $p_T > 20$ ,  $|\eta| < 2.4$ , RelIso POG, tightIP2D, SIP3D > 4  
        // Tight Muon ID, RelIso POG, tightIP2D, SIP3D > 4  
        if(lep.isMuon){  
            passId = getMuonId(iTight);  
        }  
    }  
}
```

Lepton Selector

What Lepton Selector gives back to you:

```
// Set params for the next selectors  
SetParam("selLeptons", selLeptons );  
SetParam("vetoLeptons", vetoLeptons);  
SetParam("looseLeptons", looseLeptons);  
SetParam("genLeptons", genLeptons );
```

These are std vector of 'Lepton' passing the corresponding definitions.

Also gen leptons (including leptons from taus).

Jet Selector

Define your working points for b-tagging and pt and eta of your selected and veto jets.

```
if      (gSelection == iStopSelec || iTopSelec) stringWP = "Medium";
else if (gSelection == iWWSelec)    stringWP = "Loose";
else                                     stringWP = "Medium";

if      (gSelection == iStopSelec || iTopSelec){
    jet_MaxEta = 2.4;
    jet_MinPt  = 30;
    vetoJet_minPt = 20;
}
else if(gSelection == iWWSelec){
    jet_MaxEta = 4.7;
    jet_MinPt  = 30;
    vetoJet_minPt = 20;
}
else{
    jet_MaxEta = 2.4;
    jet_MinPt  = 30;
    vetoJet_minPt = 20;
}
```

Some analysis use more than one WP for btagging → this is going to be implemented soon.

Jet Selector

What Jet Selector gives back to you:

```
SetParam("selJets",    selJets);  
SetParam("Jets15",     Jets15);  
SetParam("vetoJets",   vetoJets);  
SetParam("genJets",    genJets);
```

These are std vector of 'Jet' passing the corresponding definitions. B-tagging information and SFs are inside the jets.

Jets15 → Jets with $pt > 15$ GeV. Used for systematic uncertainties!

Event Builder

Triggers, PU reweighting and uncertainties, MET filters, channel definitions, etc.

```
Bool_t EventBuilder::PassesDoubleElecTrigger(){
    if(gIsFastSim) return true; // no trigger in FastSim samples
    Bool_t pass = false;
    if(gSelection == iStopSelec || gSelection == iTopSelec || gSelection == iTWSelec || gSelection == iWWSelec)
        pass = (
            Get<Int_t>("HLT_BIT_HLT_Ele23_Ele12_CaloIdL_TrackIdL_IsoVL_DZ_v") ||
            Get<Int_t>("HLT_BIT_HLT_Ele17_Ele12_CaloIdL_TrackIdL_IsoVL_DZ_v") ||
            Get<Int_t>("HLT_BIT_HLT_DoubleEle33_CaloIdL_GsfTrkIdVL_MW_v") ||
            Get<Int_t>("HLT_BIT_HLT_DoubleEle33_CaloIdL_GsfTrkIdVL_v") );

    return pass;
}

Bool_t EventBuilder::PassesDoubleMuonTrigger(){
    if(gIsFastSim) return true; // no trigger in FastSim samples
    Bool_t pass = false;
    if(gSelection == iStopSelec || gSelection == iTopSelec || gSelection == iTWSelec || gSelection == iWWSelec)
        pass = (
            Get<Int_t>("HLT_BIT_HLT_Mu17_TrkIsoVVL_Mu8_TrkIsoVVL_DZ_v") ||
            Get<Int_t>("HLT_BIT_HLT_Mu17_TrkIsoVVL_Mu8_TrkIsoVVL_v") ||
            Get<Int_t>("HLT_BIT_HLT_Mu17_TrkIsoVVL_TkMu8_TrkIsoVVL_v") ||
            Get<Int_t>("HLT_BIT_HLT_Mu17_TrkIsoVVL_TkMu8_TrkIsoVVL_DZ_v") );

    return pass;
}
```

Define here your different trigger paths.

Event Builder

```
// >>>>>>> Calculate PU weight and variations
```

```
if( !gIsData){
```

[illegible]

```
//PUSF      = fPUWeight      ->GetWeight(nTrueInt);
```

```
//PUSF_Up    = fPUWeightUp    ->GetWeight(nTrueInt);
```

```
//PUSF_Down = fPUWeightDown->GetWeight(nTrueInt);
```

```
PUSF      = 1;
```

```
PUSF_Up = 1;
```

```
PUSF_Down = 1;
```

```
} else{
```

```
PUSF      = 1;
```

```
PUSF_Up = 1;
```

```
PUSF_Down = 1;
```

}

[illegible]

```
// ### 2 LEPTONS
```

```
TriggerSF = 1; TriggerSF Up = 1; TriggerSF Down = 1;
```

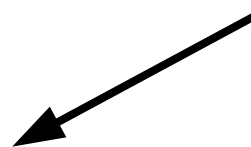
```
/*if(selLeptons.size() < 2) continue; // At least 2 selected leptons
```

```
if (selLeptons[0].IsMuon && selLepton[1].IsMuon) qChannel = iMuon
```

```
else if(selLeptons[0].IsElec && selLepton[1].IsElec) qChannel = iElec;
```

```
else qChannel = iElMu;
```

PU weights and Trigger SF coming soon!



This is going to be generalized!

'enum' for channels in 'Functions' package:
possibility of adding **trilepton** channels, etc.

Event Builder

What Jet Selector gives back to you:

```
// Set Params to pass all the info...  
SetParam("TriggerSF",      TriggerSF);  
SetParam("TriggerSF_Up",   TriggerSF_Up);  
SetParam("TriggerSF_Down", TriggerSF_Down);  
SetParam("PUSF",          PUSF);  
SetParam("PUSF_Up",       PUSF_Up);  
SetParam("PUSF_Down",     PUSF_Down);  
  
SetParam("gChannel",      gChannel);  
SetParam("NormWeight",    NormWeight);  
SetParam("passTrigger",   passTrigger);  
SetParam("isSS",          isSS);  
SetParam("METfilters",    METfilters);
```

Your analysis

The next (and last) step depends on the analysis: each analysis has its own package to perform the **event selection**.

You can implement your selection as you want! Anyway the plotter is prepared for running on **mini trees**: this is the recommendation. If you have control regions very different to your baseline selection there are several options to proceed.

- Create different mini trees for different control regions.
- Create both histograms (probably better for control regions) and minitrees (probably better for baseline or signal regions).

Mini trees are specially useful for:

- Signal region optimization.
- Add or remove cuts, see the effect of different cuts.
- Input of a BDT or other MVA method.
- Changing bins and style of the histograms

For a well defined **control region**, as the Z peak, where the number of events is huge, we recommend to **create histograms**.

Your analysis

Your selector must be in a new folder in 'packages'. You can start for copying another analysis (top, stop, WW...).

There's currently an example of selector that produces a mini tree with a default selection: [CreateMiniTree](#). You can use this base. Basically:

Declare everything (tree variables, histograms...) in the `Selector.h`. In the `Selector.C`:

- **Initialise()**: In this function initialize every variable, tree and histogram. Use [CreateTree](#), [CreateH1F](#)...
Also, use [GetParam<>\(\)](#) to recover all the parameters (vectors of selected leptons and jets, triggers...) from all the selectors.
- **Loop()**: Implement here your selection/skim and fill your histograms and mini trees.

Run the analysis

Use the **shared script** 'RunAnalysis.C'.

Inputs:

- The sample to analyze.
- The analysis to run.
- Running parameters: number of slots, number of events...
- Extra parameters: some analysis need more inputs (stop mass, etc).

It uses the **spread sheet** to search for the samples and cross sections, but calculates the weight counting the number of events in the sample.

If our parameter is **sample**, trees are named:

"Tree_" + **sample** + _[number] + ".root"

Output tree is typically named "Tree_" + **sample** + ".root". In a folder named [Analysis]_temp/.

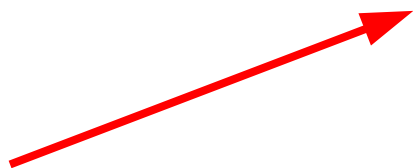
Run the analysis

```
R__LOAD_LIBRARY(DatasetManager/DatasetManager.C+)
#include "DatasetManager/DatasetManager.h"
#include "TLorentzVector.h"

Bool_t IsMCatNLO(TString sampleName);
void GetCount(vector<TString> Files, Bool_t IsData = false);
void RunAnalyserPAF(TString sampleName = "TTbar_Powheg", TString Selection = "StopDilep", Int_t nSlots
Float_t GetSMSnorm(Int_t mStop, Int_t mLsp);
Double_t GetStopXSec(Int_t StopMass);

vector<TString> Files;
Double_t SumOfWeights;
Long64_t Count;
Long64_t nTrueEntries;
Float_t xsec;
Bool_t verbose = true;

enum          sel          {iStopSelec, iTopSelec, iTWSelec, iWWSelec, ittDMSelec, ittHSelec, nSel};
const TString tagSel[nSel] = {"Stop", "          Top",      "TW",      "WW",      "ttDM",      "ttH"      };
```



Add here your analysis.

Run the analysis

Add your selector! First, LeptonSelector, JetSelector and EventBuilder are executed.

```
// Name of analysis class
//-----
myProject->AddSelectorPackage("LeptonSelector");
myProject->AddSelectorPackage("JetSelector");
myProject->AddSelectorPackage("EventBuilder");
if      (sel == iStopSelec) myProject->AddSelectorPackage("StopAnalysis");
else if (sel == iTopSelec ) myProject->AddSelectorPackage("TopAnalysis");
else if (sel == iWWSelec  ) myProject->AddSelectorPackage("WWAnalysis");
else                                     myProject->AddSelectorPackage("CreateMiniTree");

// Additional packages
//-----
myProject->AddPackage("Lepton");
myProject->AddPackage("Jet");
myProject->AddPackage("mt2");
myProject->AddPackage("Functions");
myProject->AddPackage("LeptonSF");
myProject->AddPackage("BTagSFUtil");
myProject->AddPackage("PUWeight");
```

Include the packages that you use in the analysis. Most of them are mandatory (used by LeptonSelector, etc.).

Run the analysis

Examples of use:

- `RunAnalysis("TTbar_Powheg", "tW", 20)`

Your analysis

Name of the sample

Number of slots

- `RunAnalysis("WW & WW_ext && WW_ext2", "tW", 20)`

Several input samples (same process), one output.

- `RunAnalysis("ZZ", "Top", 20, 1000)`

Only analyses the first 1000 events

- `RunAnalysis("MuonEG", "StopDilep", 20, -4)`


Divides the sample in 4 parts (you can use -n). Then it does the merge. It's useful as PAF have problems when the output is very big... (work in progress).

- `RunAnalysis("LocalFile:~juanr/Tree_WZ.root", "StopDilep", 1)`

Gets a local file! (To be checked and updated for a general case).

Plotter

The “Plotter” environment is prepared to share some common tools, but you can use your own stuff to draw your histos or process your mini trees.

 [Stop2L](#) Common classes to easily make plots from mini trees: [Plot.C](#), [Histo.C](#). (Under development). Also create datacards, print yields, etc.

 [WW](#)

 [Histo.C](#)

 [Histo.h](#)

 [Plot.C](#)

More tools to compare histograms, dump event variables, etc, coming soon!

 [Plot.h](#)

Each analysis can have a folder for it's own stuff. No object on these folders are ever going to be revised!

Plotter (example)

The plotter is quite preliminary (these slides will be updated!) but this is an example of how to use the classes with mini trees:

```
void Draw(TString var, TString cuts, TString chan, Int_t nbins, Double_t x0, Double_t xN, TString xlabel, TString name){  
    Plot* p = new Plot(var, cuts, chan, nbins, x0, xN, "Title", xlabel);  
    //p->AddSample("WJetsToLNu_aMCatNLO", "WJets", itBkg, kGreen-3, 0.50);  
    p->SetPath(path);  
}
```

Create your plot:

```
Plot* p = new Plot(var, cuts, chan, nbins, bin0, binN, title, xlabel)
```

Example:

```
Plot("TMET", "TNJets > 1 && TNBJets > 0", "ElMu", 30, 0, 300, "MET", "MET [GeV]")
```

Set path and name of the mini trees (inside the sample.root):

```
p->SetPath("~/AnalysysPAF/Stop_temp/")  
p->SetTreeName("tree");
```

Plotter (example)

Add the samples:

AddSample("Name_of_minitree", "Process", itType, Color, NormUnc)

- **Process** → To group backgrounds... With a **Color**
- **itType** → itBkg, itSignal, itData, itSys.
- **NormUnc** → norm. uncertainty for this process on the datacards.

Example:

```
p->AddSample("WZ", "VV", itBkg, kYellow-10, 0.50);  
p->AddSample("WW", "VV", itBkg);  
p->AddSample("ZZ", "VV", itBkg);  
AddSample("WJetsToLNu_MLM", "WJets", itBkg, kGray, 0.5);  
p->AddSample("TTWToLNu", "ttV", itBkg, kOrange-3, 0.5);  
p->AddSample("TTWToQQ", "ttV", itBkg);  
p->AddSample("TTZToQQ", "ttV", itBkg);  
p->AddSample("TTZToLLNuNu", "ttV", itBkg);
```

Plotter (example)

Add the systematics and plot, produce datacards or whatever.

```
p->AddSystematic("JES");  
p->AddSystematic("LepEff");  
p->AddSystematic("Btag");  
p->AddSystematic("MisTag");  
  
p->PrintYields();  
p->doSetLogy = false;  
p->DrawStack("0", 1);  
p->doSetLogy = true;  
p->DrawStack("0_log", 1);  
//p->PrintSystYields();  
p->MakeAllDatacards();  
delete p;
```