Implementing Cuts in MadAnalysis5

Dipan Sengupta

Michigan State University
East Lansing, U.S.A
What is a cut, and why cut

• A “cut” is a choice of a value of a discriminating observable between two distributions.
• Let’s call them signal and background.
• The objective is to optimize S/B by choosing a value of an observable (a cut).

• Cuts on observables can be simple: like a simple cut on the transverse momenta $p_T$
• Cuts on observables can be complex: A function of other objects $M_{T_2}$
A Toy BSM Case

• Toy example: A SUSY process with leptons, jets and missing transverse energy

**ttbar as a SUSY background**

For a ttbar event to produce large MET (100-200 GeV), need at least one $W \rightarrow l \nu, l = e, \mu, \tau$.

→ reduces Njets.

Searches for SUSY in hadronic channels benefit from lepton vetoes.

τ-leptons are a problem!
A Toy BSM Case

![Graph showing production cross section in GeV and pb for SUSY events with various decay modes.](image)

![Graph showing CMS 7 TeV measurements and 8 TeV theory predictions for W, Z, WY, WZ, and ZZ production cross sections.](image)

**WHAT EXACTLY AM I SUPPOSED TO DO WITH THIS GUT?**
Examples of Cuts: Start from Isolation

Lepton Isolation

- QCD backgrounds produce leptons inside jets
- Requiring isolated leptons strongly rejects these QCD backgrounds
- Sum the energy deposited within $\Delta R < 0.3$
  - Pile-up (PU) corrections are applied
  - Charged hadron PU already removed with vertex cuts
  - Neutral hadron pile-up approximated as $0.5 \times$ charged hadron PU and is called $\Delta \beta$

$$I_{rel} = \frac{\Sigma p_T(\text{charged}) + \max[\Sigma E_T(\text{neutral}) + \Sigma E_T(\text{photon}) - \Delta \beta, 0]}{p_T(\mu \text{ or } e)}$$

Taejong Kim and Sam Bein’s Talk
Examples of Cuts: Simple Cuts

Start with simple cuts

1. In SUSY with a heavy decaying particle jets and leptons have large momenta compared to SM.
2. Large missing transverse energy in SUSY

\[ H_T = \sum p_T (\text{jet}) \]
Examples of Cuts: Complex Cuts I

Simple cuts may not be enough to suppress backgrounds

\[ \alpha_T = \frac{1}{2} \frac{H_T - \Delta H_T}{\sqrt{H_T^2 - (MHT)^2}} \]

\[ H_T = \sum_{\text{jets } j} p_T^j \]

\[ \Delta H_T = p_{T\text{pseudojet 1}} - p_{T\text{pseudojet 2}} \]

\( \alpha_T \) variable is designed to separate events with low MET or mismeasurement from genuine events.

If \( N_{\text{jets}} > 2 \), jets are merged into 2 pseudojets (minimizing the \( \Delta E_T \) between them)

Randall, Tucker Smith 2009
Examples of Cuts: Complex cuts II

More complex cuts: Transverse mass cuts

\[ m_T = \sqrt{2(\vec{p}_T^{\text{lepton}}E_T^{\text{miss}} - \vec{p}_{\text{lepton}} \cdot \vec{E}_{T}^{\text{miss}})} \]

Originally used to discover W boson in the 1980's
Examples of Cuts: Complex Cuts III

More complex cuts: Transverse mass cuts

What if there are two/multiple branches of visible and missing energy

\[ M_{T2}(m_c) = \min_{p_T^{c(1)} + p_T^{c(2)} = p_T^{miss}} \left[ \max \left( m_T^{(1)}, m_T^{(2)} \right) \right] \]

End-point corresponds to the mass of the parent particle if mass of invisible particle is known

Definition of MT2

- A pair of semi-invisibly decaying particles

The Best thing that we have: Missing ET constraint

\[ E_T = \vec{q}_1 + \vec{q}_2 = - (\vec{p}_1 + \vec{p}_2) \]

- If \( \vec{q}_1 \) and \( \vec{q}_2 \) are obtainable: \( M_W^2 \geq \max \{ m_T^2(\vec{p}_1, \vec{q}_1), m_T^2(\vec{p}_2, \vec{q}_2) \} \)
- But since we don’t get them, at most we can do:

\[ M_W^2 \geq m_{T2}^2 \equiv \min_{q_1 \neq q_2 = \vec{q}_T} \left[ \max \left( m_T^2(\vec{p}_1, \vec{q}_1), m_T^2(\vec{p}_2, \vec{q}_2) \right) \right] \]
Razor variables approximate boosted frames with a razor frame, where visible energies are written as a scale invariant under longitudinal boosts.

Razor boost: \( \beta_L^R \equiv \frac{p_{z1}^j + p_{z2}^j}{E_{j1} + E_{j2}} \)

Scale: \( M_R \equiv \sqrt{(E_{j1} + E_{j2})^2 - (p_{z1}^j + p_{z2}^j)^2} \)

A transverse observable \( M_T^R \) is also defined, whose maximum value peaks at \( M_\Delta \):

\[
M_T^R \equiv \sqrt{\frac{E_T^{miss}(p_T^{j1} + p_T^{j2}) - E_T^{miss} (\vec{p}_T^{j1} + \vec{p}_T^{j2})}{2}}
\]

The ratio of these two quantities gives a dimensionless discriminant, the Razor \( R \):

\[
R \equiv \frac{M_T^R}{M_R}
\]

Objects are merged in 2 pseudojets, with hemisphere algorithm

C. Rogan, 2011
Cuts can also exploit the shape of particles/jets.

\[
T_{\perp,g} \equiv \max_{\vec{n}_T} \frac{\sum_i |\vec{p}_{\perp,i} \cdot \vec{n}_T|}{\sum_i p_{\perp,i}}
\]

M. Guchait, D.S., 2011
Substructures help background reduction + classification of jets

Classic Example $H \rightarrow b\bar{b}$

Find local subclusters of energy within a jet

Step through clustering history to identify a hard splitting

Remove UE/Pile up contamination through pruning/filtering

Related ideas

N-Subjettiness -> Quasi-minimization of N-point sub-structure

Q-Jets -> Tree based substructure to reduce fluctuations in the pruned jet mass

Kaplan, Tweedie, 2009

$R_{b\bar{b}} \sim \frac{1}{\sqrt{z(1-z)}} \frac{m_H}{p_T}$

“BDRS” 0802.4280,
Examples of Cuts: Boosted Objects

BDRS tagger helps in resolving the mass peak better, removes contamination from ISR

Ghosh, Guchait, D.S 2012

Top Mass reconstruction using JH tagger
HOW TO IMPLEMENT CUTS IN MADANALYSIS5
How to Implement Cuts: Normal Mode

Importing data sets

```plaintext
ma5> import tt*.lhe as ttbar
ma5> import Wj*.lhe as Wjets
```

Assigning a type for data sets

```plaintext
ma5> set ttbar.type = signal
ma5> set Wjets.type = background
```

More options are available:

- Specifying the histogram binning
  ```plaintext
  ma5> plot MET 100 0 1000
  ma5> plot M(mu+ mu-)
  ```

- Specifying other options
  ```plaintext
  ma5> plot PT(mu) [logY]
  ma5> plot MET 100 0 1000 [normalize2one]
  ```
How to Implement Cuts and Histos: Normal Mode

**Histograms**

- Observable can be related to the event or the properties of a particle
- Plethora of observables: \( N, E, ET, M, MT, P, PT, PX, PY, PZ, \theta, \eta, \ldots, \alpha_T \)
- Including sophisticated observables: \( \alpha_T, MT^2, MT^2W \)

```
ma5> plot MET 100 0 1000
ma5> plot MET
ma5> plot PT(mu)
```

More options are available:

- Specifying the histogram binning
  ```
  ma5> plot MET 100 0 1000
  ```
- Specifying other options
  ```
  ma5> plot PT(mu) [logY]
  ma5> plot MET 100 0 1000 [normalize2one]
  ```
Defining an analysis: plots and/or cuts

- **Cuts**: selecting / rejecting events

```
ma5> reject MHT < 50
ma5> select N(mu) >= 2
```

- **Cuts**: selecting / rejecting a particle or a combination

```
ma5> select (mu) PT > 50
ma5> select 80 < M (mu+ mu-) < 100
```

Launching the Analysis

* SampleAnalyzer 2.0 for MadAnalysis 5 - Welcome.
* Option choices: selecting analysis = 'MadAnalysis5job'.
* Extracting the following sample files:
* 1/4 ~/samples/ttbar_sl_1.lhe.gz
  => file size : 107.09 Mo
  => sample produced by MadGraph.
  => progress [==========================================]
Generate the HTML report

**MadAnalysis 5 report**

*Created by econte on 05 November 2012, 21:29:45*

---

**Setup**

**Command history**

- `ma5>define mu = mu+ mu-
- `ma5>import samples/ttbar_sl_1.lhe.gz
- `ma5>import samples/ttbar_sl_2.lhe.gz
- `ma5>import samples/ttbar_fh.lhe.gz
- `ma5>import samples/zz.lhe.gz
- `ma5>ma5>plot MET
- `ma5>ma5>plot PT(mu) 20 0 100
- `ma5>ma5>reject MET > 100
- `ma5>ma5>reject (mu) PT < 20
- `ma5>ma5>plot M(mu+ mu-) 20 0 100
- `ma5>ma5>submit
- `ma5>plot MET
- `ma5>plot PT(mu) 20 0 100
- `ma5>reject MET > 100
- `ma5>reject (mu) PT < 20
- `ma5>plot M(mu+ mu-) 20 0 100
- `ma5>submit

---

**Configuration**
Look at all the signal regions

Multiple Signal regions with overlapping cuts
How to Implement Histos and Cuts: Expert Mode

Available Information

Exclusion Curves

Representative cutflows

**Table:**

<table>
<thead>
<tr>
<th>Selection</th>
<th>(200/100) GeV</th>
<th></th>
<th>(500/150) GeV</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>all events</td>
<td>3630</td>
<td>-</td>
<td>115.79</td>
<td>-</td>
</tr>
<tr>
<td>3 tight e, µ or τh</td>
<td>482.20</td>
<td>13.3%</td>
<td>18.06</td>
<td>15.6%</td>
</tr>
<tr>
<td>4th lepton veto</td>
<td>481.49</td>
<td>99.9%</td>
<td>18.03</td>
<td>99.8%</td>
</tr>
<tr>
<td>conversions and low-mass veto</td>
<td>463.71</td>
<td>96.3%</td>
<td>17.79</td>
<td>98.6%</td>
</tr>
<tr>
<td>b-jet veto</td>
<td>456.68</td>
<td>98.5%</td>
<td>17.47</td>
<td>98.2%</td>
</tr>
<tr>
<td>$\not{p}_T &gt; 50$ GeV</td>
<td>317.00</td>
<td>69.4%</td>
<td>16.98</td>
<td>97.2%</td>
</tr>
<tr>
<td>$M_T &gt; 100$ GeV</td>
<td>111.97</td>
<td>35.3%</td>
<td>12.74</td>
<td>75.1%</td>
</tr>
<tr>
<td>$M_{\ell\ell} &gt; 75$ GeV</td>
<td>103.49</td>
<td>92.4%</td>
<td>11.71</td>
<td>91.9%</td>
</tr>
</tbody>
</table>

**Graph:**

CMS Preliminary 35.9 fb⁻¹ (13 TeV)

- Observed ± 1σ_{theory}, NLO-NLL excl.
- Expected ± 1σ_{experiment}

The figure shows the exclusion curves for the mass difference between the neutralino $\tilde{\chi}_1^0$ and chargino $\tilde{\chi}_2^0$, with cuts on various kinematic variables.
How to Implement Histos and Cuts: Expert Mode

Handling Cuts

Cuts for all signal regions: Isolation, Trigger, Pre-selection Criteria

Some signal region cuts work for multiple Signal regions

Some signal region cuts work for specific signal Regions
Handling cuts and histograms

- Naive approach **not efficient** (see cut #4 for instance)

```plaintext
count the event in region D
if (condition 3)
{
    count the event in region C
    if (condition 4)
    {
        count the event in region A
    }
}
if (condition 4)
{
    count the event in region B
}
```

- **A more efficient** algorithm has been implemented
  - Each cut condition is only evaluated once
  - It is applied to all surviving regions **simultaneously**
The analyzer code in MA5 expert mode consists of the basic functions:

- **Initialize**: Initialization of signal regions, declaration of cuts, histograms and user defined functions.
- **Execute**: Containing the analysis cuts and weights applied to each event.
- **Finalize**: Production of the results, histograms and cut flows.
Step 1: Declare all the signal regions in the analysis

// =====Declare the signal regions in this analysis=====
Manager()->AddRegionSelection("2jloose");
Manager()->AddRegionSelection("4jloose");
Manager()->AddRegionSelection("7jloose");
Manager()->AddRegionSelection("2bloose");
Manager()->AddRegionSelection("3bloose");
Manager()->AddRegionSelection("7j3bloose");
//Manager()->AddRegionSelection("EM7");
//===================================//
Step 2: Declare all cuts that are not region specific

// Declare all the cuts used in the analysis.
// Note that the order in which you declare cuts defines the order in which
// they appear in the cut flow table. What you want this order to be is the
// order in which the cuts are applied. It's therefore on you to declare cuts
// in the order that they are applied, so as to have a sensible cut flow at
// the end.
// Cuts are declared with RSManager's AddCut method. A cut needs a (string)
// name, and a set of signal regions which it applies to; if the latter is not
// specified, i.e. AddCut is called with only a name as its argument, the cut
// is taken to apply to *all* signal regions, i.e. it's a common cut, not a
// region-specific cut. We do this here:

//===General Cuts===================
Manager()->AddCut("MET > 30 GeV");
Manager()->AddCut("HT > 1000 GeV");
Manager()->AddCut("N(j) >= 1");
Manager()->AddCut("MT2 > 200 GeV");
Manager()->AddCut("dphij1met > 0.3");
Manager()->AddCut("dphij2met > 0.3");
Manager()->AddCut("dphij3met > 0.3");
Manager()->AddCut("dphij4met > 0.3");
Manager()->AddCut("ratio < 0.5");
Manager()->AddCut("ElectronVeto");
Manager()->AddCut("MuonVeto");

Manager()->AddCut("Leading jet pT > 250 GeV");
Step 3: Declare all cuts that are region specific

// Some cuts, on the other hand, apply only to some of the regions and not all // of them. To declare such cuts, call the method AddCut with a second // argument - an array of strings - with each string corresponding to the name // of a signal region. This is what we do next. // (Note that we have declared all common cuts first, then all region-specific // cuts afterwards; we clarify that it doesn't need to be done like this - // that's just the order of the cuts in the experimental analysis we're // implementing here.)

//===Signal Region Cuts====RegionEMs==========
    Manager()->AddCut("HT > 1000 GeV", HT1000);
Manager()->AddCut("HT > 1500 GeV", HT1500);
Manager()->AddCut("MT2 > 400 GeV", MT2400);
Manager()->AddCut("MT2 > 600 GeV", MT2600);
Manager()->AddCut("MT2 > 800 GeV", "7jtight");
Manager()->AddCut("MT2 > 1000 GeV", "4jloose");
Manager()->AddCut("MT2 > 1200 GeV", "2jloose");
Manager()->AddCut("MT2 > 1400 GeV", MT21400);
Manager()->AddCut("N(j) >= 2", nj2);
Manager()->AddCut("N(j) >= 4", nj4);
Manager()->AddCut("N(j) >= 7", nj7);
Manager()->AddCut("N(b) >= 2", nb2);
Manager()->AddCut("N(b) >= 3", nb3);
Declare all the Histograms in the analysis
Follows the same logic as cuts

Manager()->AddHisto("MT",10,0,300); // Fig. 2a
Manager()->AddHisto("MET",10,100,350); // Fig. 2b
Manager()->AddHisto("MT2W",17,75,500); // Fig. 2c
Manager()->AddHisto("chi2",20,0,20); // Fig. 2d
Manager()->AddHisto("HTratio",25,0,1); // Fig. 2e
Manager()->AddHisto("pT(leading b-jet)",9,30,300); // Fig. 2g
Manager()->AddHisto(\"pT(l)\",8,20,100); // Fig. 2i
Manager()->AddHisto("JetMETMindeltaPhi",15,0,\Pi); // Fig. 2f
Manager()->AddHisto("deltaR(l, leading b-jet)\",15,0,5); // Fig. 2h
Manager()->AddHisto("MT2W after \( MT>120 \)\",17,75,500); // Fig. add 12
Manager()->AddHisto("chi2 after \( MT>120 \)\",20,0,20); // Fig. add 14
Manager()->AddHisto("HTratio after \( MT>120 \)\",25,0,1); // Fig. add 16
Manager()->AddHisto("JetMETMindeltaPhi after \( MT>120 \)\",25,0,\Pi); // Fig. add 17
Manager()->AddHisto("pT(leading b-jet) after \( MT>120 \)\",20,0,500); // Fig. add 13
Manager()->AddHisto("deltaR(l, leading b-jet) after \( MT>120 \)\",20,0,5); // Fig. 
          // add 15
Define Objects

//Defining the containers
vector<const RecJetFormat*>& SignalJets, myjet, pseudojets;
vector<const RecLeptonFormat*>& SignalMuons, SignalElectrons;
vector<const RecTauFormat*>& SignalTaus;

//electron with pt>20
for(unsigned int ie=0; ie<event.rec()->electrons().size(); ie++)
    {
        const RecLeptonFormat * CurrentElectron = &(event.rec()->electrons()[ie]);
        double pt = CurrentElectron->pt();
        double abseta = fabs(CurrentElectron->eta());
        if(pt > 20. && abseta < 2.47)
            SignalElectrons.push_back(CurrentElectron);
    }

for(unsigned int ij=0; ij<event.rec()->jets().size(); ij++)
    {
        const RecJetFormat * CurrentJet = &(event.rec()->jets()[ij]);
        if ( CurrentJet->pt() > 30.0 && abs(CurrentJet->eta())<2.8){
            SignalJets.push_back(CurrentJet);
            HT = HT + CurrentJet->pt();
            px = px + CurrentJet->px();
            py = py + CurrentJet->py();
            pz = pz + CurrentJet->pz();
            nj = nj + 1;
            if (CurrentJet->btag()) nbjets = nbjets+1;
        }
Define Objects

//Defining the containers
vector<RecJetFormat*> SignalJets, myjet, pseudojets;
vector<RecLeptonFormat*> SignalMuons, SignalElectrons;
vector<RecTauFormat*> SignalTaus;

//electron with pt>20
for(unsigned int ie=0; ie<event.rec()->electrons().size(); ie++)
{
  const RecLeptonFormat * CurrentElectron = &event.rec()->electrons()[ie];
  double pt = CurrentElectron->pt();
  double abseta = fabs(CurrentElectron->eta());
  if(pt > 20. && abseta < 2.47)
    SignalElectrons.push_back(CurrentElectron);
}

for(unsigned int ij=0; ij<event.rec()->jets().size(); ij++)
{
  const RecJetFormat * CurrentJet = &event.rec()->jets()[ij];
  if ( CurrentJet->pt() > 30.0 && abs(CurrentJet->eta())<2.8) {
    SignalJets.push_back(CurrentJet);
    HT = HT + CurrentJet->pt();
    px = px + CurrentJet->px();
    py = py + CurrentJet->py();
    pz = pz + CurrentJet->pz();
    nj = nj + 1;
    if (CurrentJet->btag()) nbjets = nbjets+1;
  }
Define Objects

//Defining the containers
vector<const RecJetFormat*>& SignalJets, myjet, pseudojets;
vector<const RecLeptonFormat*>&SignalMuons,SignalElectrons;
vector<const RecTauFormat*>&SignalTaus;

//electron with pt>20
for(unsigned int ie=0; ie<event.rec()->electrons().size(); ie++)
{
    const RecLeptonFormat * CurrentElectron = &event.rec()->electrons()[ie];
    double pt = CurrentElectron->pt();
    double abseta = fabs(CurrentElectron->eta());
    if(pt > 20. && abseta < 2.47)
        SignalElectrons.push_back(CurrentElectron);
}

for(unsigned int ij=0; ij<event.rec()->jets().size(); ij++)
{
    const RecJetFormat * CurrentJet = &event.rec()->jets()[ij];
    if((CurrentJet->pt() > 30.0 && abs(CurrentJet->eta())<2.8)){
        SignalJets.push_back(CurrentJet);
        HT = HT + CurrentJet->pt();
        px = px + CurrentJet->px();
        py = py + CurrentJet->py();
        pz = pz + CurrentJet->pz();
        nj = nj + 1;
        if(CurrentJet->btag()) nbjets = nbjets+1;
}
Define Objects

//Defining the containers
vector< const RecJetFormat >* SignalJets, myjet, pseudojets;
vector< const RecLeptonFormat >* SignalMuons, SignalElectrons;
vector< const RecTauFormat >*SignalTaus;

//electron with pt > 20
for (unsigned int ie = 0; ie < event.rec()->electrons().size(); ie++)
{
  const RecLeptonFormat * CurrentElectron = &(event.rec()->electrons()[ie]);
  double pt = CurrentElectron->pt();
  double abseta = fabs(CurrentElectron->eta());
  if (pt > 20. && abseta < 2.47)
    SignalElectrons.push_back(CurrentElectron);
}

for (unsigned int ij = 0; ij < event.rec()->jets().size(); ij++)
{
  const RecJetFormat * CurrentJet = &(event.rec()->jets()[ij]);
  if (CurrentJet->pt() > 30.0 && abs(CurrentJet->eta()) < 2.8){
    SignalJets.push_back(CurrentJet);
    HT = HT + CurrentJet->pt();
    px = px + CurrentJet->px();
    py = py + CurrentJet->py();
    pz = pz + CurrentJet->pz();
    nj = nj + 1;
    if (CurrentJet->btag()) nbjets = nbjets+1;
}

Basic cuts on Electrons
```cpp
// Defining the containers
vector<const RecJetFormat*> SignalJets, myjet, pseudojets;
vector<const RecLeptonFormat*> SignalMuons, SignalElectrons;
vector<const RecTauFormat*> SignalTaus;

// Electron with pt>20
for(unsigned int ie=0; ie<event.rec()->electrons().size(); ie++)
{
    const RecLeptonFormat * CurrentElectron = &(event.rec()->electrons()[ie]);
    double pt = CurrentElectron->pt();
    double abseta = fabs(CurrentElectron->eta());
    if(pt > 20. && abseta < 2.47) SignalElectrons.push_back(CurrentElectron);
}

// Jet with pt>30
for(unsigned int ij=0; ij<event.rec()->jets().size(); ij++)
{
    const RecJetFormat * CurrentJet = &(event.rec()->jets()[ij]);
    if( CurrentJet->pt() > 30.0 && abs(CurrentJet->eta())<2.8) { SignalJets.push_back(CurrentJet);
        HT = HT + CurrentJet->pt();
        px = px + CurrentJet->px();
        py = py + CurrentJet->py();
        pz = pz + CurrentJet->pz();
        nj = nj + 1;
        if (CurrentJet->btag()) nbjets = nbjets+1;
    }
}
```
Define Objects

// Defining the containers
vector<const RecJetFormat*> SignalJets, myjet, pseudojets;
vector<const RecLeptonFormat*> SignalMuons, SignalElectrons;
vector<const RecTauFormat*> SignalTaus;

// Electron with pt>20
for(unsigned int ie=0; ie<event.rec()->electrons().size(); ie++)
{
    const RecLeptonFormat * CurrentElectron = &(event.rec()->electrons()[ie]);
    double pt = CurrentElectron->pt();
    double absEta = fabs(CurrentElectron->eta());
    if (pt > 20. && absEta < 2.47)
        SignalElectrons.push_back(CurrentElectron);
}

for(unsigned int ij=0; ij<event.rec()->jets().size(); ij++)
{
    const RecJetFormat * CurrentJet = &(event.rec()->jets()[ij]);
    if (CurrentJet->pt() > 30.0 && abs(CurrentJet->eta())<2.8)
    
    SignalJets.push_back(CurrentJet);
    HT = HT + CurrentJet->pt();
    px = px + CurrentJet->px();
    py = py + CurrentJet->py();
    pz = pz + CurrentJet->pz();
    nj = nj + 1;
    if (CurrentJet->btag()) nbjets = nbjets+1;
Define Objects

```cpp
//Defining the containers
vector<const RecJetFormat*> SignalJets, myjet, pseudojets;
vector<const RecLeptonFormat*>SignalMuons,SignalElectrons;
vector<const RecTauFormat*>SignalTaus;
```

//electron with pt>20
for(unsigned int ie=0; ie<event.rec()->electrons().size(); ie++)
    {
    const RecLeptonFormat * CurrentElectron = &(event.rec()->electrons()[ie]);
    double pt = CurrentElectron->pt();
    double abseta = fabs(CurrentElectron->eta());
    if(pt > 20.0 && abseta < 2.47)
        SignalElectrons.push_back(CurrentElectron);
}

for(unsigned int ij=0; ij<event.rec()->jets().size(); ij++)
    {
    const RecJetFormat * CurrentJet = &(event.rec()->jets()[ij]);
    if( (CurrentJet->pt()) > 30.0 && abs(CurrentJet->eta())<2.8){
        SignalJets.push_back(CurrentJet);
        HT = HT + CurrentJet->pt();
        px = px + CurrentJet->px();
        py = py + CurrentJet->py();
        pz = pz + CurrentJet->pz();
        nj = nj + 1;
        if(CurrentJet->btag()) nbjets = nbjets+1;
        }
```

Basic cuts on Electrons

Basic cuts on Jets

B-tag
Isolation + Overlap removal

```cpp
for(int im=SignalMuons.size()-1;im>=0;im--)
{
    const RecLeptonFormat * myMuon = &(event.rec()->muons()[im]);
    double mympt=myMuon->pt();
    double chargeptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
            0.,IsolationEFlow::TRACK_COMPONENT);
    double neutralptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
            0.,IsolationEFlow::NEUTRAL_COMPONENT);
    double photonptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
            0.,IsolationEFlow::PHOTON_COMPONENT);
    double ttptm=chargeptm + neutralptm + photonptm;
    if(ttptm > 0.2*mympt)
        SignalMuons.erase(SignalMuons.begin()+im);
}

//======Overlap removal of jets=========================
    SignalJets = PHYSICS->Isol->JetCleaning(SignalJets, SignalElectrons, 0.4);
    SignalJets = PHYSICS->Isol->JetCleaning(SignalJets, SignalMuons, 0.4);
```
Isolation + Overlap removal

```cpp
for(int im=SignalMuons.size()-1;im>=0;im--)
{
    const RecLeptonFormat * myMuon = &(event.rec()->muons()[im]);
    double mympt=myMuon->pt();
    double chargeptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
    0.,IsolationEFlow::TRACK_COMPONENT);
    double neutralptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
    0.,IsolationEFlow::NEUTRAL_COMPONENT);
    double photonptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
    0.,IsolationEFlow::PHOTON_COMPONENT);
    double ttptm=chargeptm + neutralptm + photonptm;
    if(ttptm > 0.2*mympt)
        SignalMuons.erase(SignalMuons.begin()+im);
}
```

'========Overlap removal of jets=================='
SignalJets = PHYSICS->Isol->JetCleaning(SignalJets, SignalElectrons, 0.4);
SignalJets = PHYSICS->Isol->JetCleaning(SignalJets, SignalMuons, 0.4);
Isolation + Overlap removal

```cpp
for(int im=SignalMuons.size()-1;im>=0;im--)
{
    const RecLeptonFormat * myMuon = &(event.rec()->muons()[im]);
    double mympt=myMuon->pt();
    double chargeptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
    0.,IsolationEFlow::TRACK_COMPONENT);
    double neutralptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4
    ,0.,IsolationEFlow::NEUTRAL_COMPONENT);
    double photonptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
    0.,IsolationEFlow::PHOTON_COMPONENT);
    double ttptm=chargeptm + neutralptm + photonptm;
    if(ttptm > 0.2*mympt)
        SignalMuons.erase(SignalMuons.begin()+im);
}
```

```cpp`
//===Overlap removal of jets========================//
    SignalJets = PHYSICS->Isol->JetCleaning(SignalJets, SignalElectrons, 0.4);
    SignalJets = PHYSICS->Isol->JetCleaning(SignalJets, SignalMuons, 0.4);
```
Isolation + Overlap removal

```cpp
for(int im=SignalMuons.size()-1;im>=0;im--)
{
    const RecLeptonFormat * myMuon = &(event.rec()->muons()[im]);
    double mympt=myMuon->pt();
    double chargeptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
    0.,IsolationEFlow::TRACK_COMPONENT);
    double neutralptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
    0.,IsolationEFlow::NEUTRAL_COMPONENT);
    double photonptm=PHYSICS->Isol->eflow->sumIsolation(myMuon,event.rec(),0.4,
    0.,IsolationEFlow::PHOTON_COMPONENT);
    double ttptm=chargeptm + neutralptm + photonptm;
    if(ttptm > 0.2*mympt)
        SignalMuons.erase(SignalMuons.begin()+im);
}
```

```cpp`
//======Overlap removal of jets===================//
SignalJets = PHYSICS->Isol->JetCleaning(SignalJets, SignalElectrons, 0.4);
SignalJets = PHYSICS->Isol->JetCleaning(SignalJets, SignalMuons, 0.4);
```
Define Missing Energy

MA LorentzVector MET = event.rec() -> MET().momentum();
double METval = MET.Pt();

Mass of a four vector created out of two four vectors

MA LorentzVector Pll = Pos_el_loose[i1] -> momentum() + Neg_el_loose[jj] -> momentum();
if (Pll.Mag() < 105.0 && Pll.Mag() > 75.0) mllossf = false;
if (Pll.Mag() < 12.0) mlllow = false;

Define Transverse Mass

mtmiss = pl3.mt_met(MET)

Define MT2

double mt2_l1l2 = 0.0;
double m_trial = 0.0;
if (TightElectrons.size() >= 2) {mt2_l1l2 = PHYSICS -> Transverse -> MT2(& (TightElectrons[0] -> momentum()), & (TightElectrons[1] -> momentum()), MET, m_trial);}
ext if (TightMuons.size() >= 2) {mt2_l1l2 = PHYSICS -> Transverse -> MT2(& (TightMuons[0] -> momentum()), & (TightMuons[1] -> momentum()), MET, m_trial);}
Angular Cuts between objects

\[ \text{DeltaPhiJet1} = \text{SignalJets}[0] \rightarrow \text{dphi}_0\pi(\text{pTmiss}) \]

Execute cuts corresponding to pre-defined cut names

```c++
// General cuts

// Missing Et cut
if (!Manager()->ApplyCut(MET > 30, "MET > 30 GeV"))
    return true;

// HT cut
if (!Manager()->ApplyCut(HT > 1000, "HT > 1000 GeV"))
    return true;

// Number of jets cut
if (!Manager()->ApplyCut(nj > 1, "N(j) >= 1"))
    return true;
```

Cut associated with all the signal regions
Execute cuts corresponding to cuts associated with signal regions

//======Signal Region Specific Cuts==============//

if(!Manager()->ApplyCut((mt2 > 400),"MT2 > 400"))
return true;
if(!Manager()->ApplyCut((mt2 > 600),"MT2 > 600"))
return true;
if(!Manager()->ApplyCut((mt2 > 800),"MT2 > 800"))
return true;
if(!Manager()->ApplyCut((mt2 > 1000),"MT2 > 1000"))
return true;
if(!Manager()->ApplyCut((mt2 > 1200),"MT2 > 1200"))
return true;
if(!Manager()->ApplyCut((mt2 > 1400),"MT2 > 1400"))
return true;
Some cuts could be part of multiple signal regions

```cpp
std::string HT1000[] = {"2jloose", "4jloose", "7jloose","2bloose", "3bloose", "7j3bloose","new"};
std::string HT1500[] = {"2jtight", "4jtight", "7jtight","2btight", "3btight", "7j3btight"};
std::string MT2400[] = { "3bloose", "7j3bloose","3btight", "7j3btight"};
std::string MT2600[] = { "7jloose","2bloose","2btight"};
std::string MT21400[] = { "2jtight","4jtight"};
std::string nj2[] =
  { "2jloose","2jtight","2bloose","2btight","3bloose","3btight"};
std::string nj4[] = { "4jloose","4jtight","new"};
std::string nj7[] = { "7jloose","7jtight","7j3bloose","7j3btight"};
std::string nb2[] = { "2bloose","2btight","new"};
std::string nb3[] = { "3bloose","3btight","7j3bloose","7j3btight"};
```
// Some histos which are defined only after the previous cut:
Manager()->FillHisto("MT2W after MT>120", mt2w);
Manager()->FillHisto("chi2 after MT>120", chi2);
Manager()->FillHisto("HTratio after MT>120", HTratio);
Manager()->FillHisto("JetMETMinDeltaPhi after MT>120", MinDeltaPhiJetMET);
Manager()->FillHisto("pT(leading b-jet) after MT>120", LeadingBjetpT);
Manager()->FillHisto("deltaR(l, leading b-jet) after MT>120", deltaRb1l);
Looking at the Output, present in the Output folder in xml format: Cutflows
Looking at the Output, present in the Output folder in xml format: Cutflows

bash-4.1$ ls
Build exclusion_CLs.py Input Output
bash-4.1$ cd Output/
bash-4.1$ ls
test.txt
bash-4.1$ cd test.txt/
bash-4.1$ ls
ATLAS_1604_07773_0 ATLAS_1604_07773_0.out test.txt.saf
bash-4.1$ cd ATLAS_1604_07773_0
bash-4.1$ ls
ATLAS_1604_07773.saf Cutflows Histograms
bash-4.1$ cd Cutflows/
bash-4.1$ ls
bash-4.1$
Looking at the Output, present in the Output folder in xml format: Cutflows

```xml
<SAFheader>
</SAFheader>

<InitialCounter>
"Initial number of events" #
19114 0 # nentries
1.911400e+04 0.000000e+00 # sum of weights
1.911400e+04 0.000000e+00 # sum of weights^2
</InitialCounter>

<Counter>
"at least 2 leptons" # 1st cut
1671 0 # nentries
1.671000e+03 0.000000e+00 # sum of weights
1.671000e+03 0.000000e+00 # sum of weights^2
</Counter>

<Counter>
"at least 2 OS leptons" # 2nd cut
157 0 # nentries
1.570000e+02 0.000000e+00 # sum of weights
1.570000e+02 0.000000e+00 # sum of weights^2
</Counter>
```
Looking at the Output, present in the Output folder in xml format: Histos

Root not required to plot, You can use your favorite tool.
How to test cutflows

For every signal region: after every cut

\[ n_s = \text{Acceptance} \times \text{efficiency} \times \text{cross section} \times \text{luminosity} \]

<table>
<thead>
<tr>
<th>cut</th>
<th>( \tilde{q} \to q\tilde{\chi}_1^0 ) (650/645) cutflow</th>
<th># events</th>
<th>relative change</th>
<th># events</th>
<th>relative change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial number of events</td>
<td>4544</td>
<td>-</td>
<td>-</td>
<td>4544</td>
<td>-</td>
</tr>
<tr>
<td>( E_T^{\text{miss}} &gt; 100 \text{ GeV} )</td>
<td>1917</td>
<td>58 %</td>
<td>2031</td>
<td>55 %</td>
<td></td>
</tr>
<tr>
<td>Trigger</td>
<td>1604</td>
<td>16 %</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Event cleaning</td>
<td>1592</td>
<td>1 %</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Lepton veto</td>
<td>1591</td>
<td>0.01 %</td>
<td>2022</td>
<td>0.4 %</td>
<td></td>
</tr>
<tr>
<td>( N_{\text{jets}} \leq 4 )</td>
<td>1492</td>
<td>6 %</td>
<td>1883</td>
<td>7 %</td>
<td></td>
</tr>
<tr>
<td>( \Delta \phi(E_T^{\text{miss}}, \text{jets}) &gt; 0.4 )</td>
<td>1409</td>
<td>5 %</td>
<td>1798</td>
<td>5 %</td>
<td></td>
</tr>
<tr>
<td>Jet Quality</td>
<td>1343</td>
<td>4 %</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( p_T^{1 &gt; 250 \text{ GeV}} )</td>
<td>435</td>
<td>67 %</td>
<td>426</td>
<td>76 %</td>
<td></td>
</tr>
<tr>
<td>( E_T^{\text{miss}} &gt; 250 \text{ GeV} )</td>
<td>404</td>
<td>7 %</td>
<td>402</td>
<td>6 %</td>
<td></td>
</tr>
<tr>
<td>M1 : ( E_T^{\text{miss}} = [250 - 300] \text{ GeV} )</td>
<td>58</td>
<td>86 %</td>
<td>57</td>
<td>86 %</td>
<td></td>
</tr>
<tr>
<td>M2 : ( E_T^{\text{miss}} = [300 - 350] \text{ GeV} )</td>
<td>65</td>
<td>84 %</td>
<td>69</td>
<td>83 %</td>
<td></td>
</tr>
<tr>
<td>M3 : ( E_T^{\text{miss}} = [350 - 400] \text{ GeV} )</td>
<td>59</td>
<td>86 %</td>
<td>57</td>
<td>86 %</td>
<td></td>
</tr>
<tr>
<td>M4 : ( E_T^{\text{miss}} = [400 - 500] \text{ GeV} )</td>
<td>85</td>
<td>79 %</td>
<td>81</td>
<td>80 %</td>
<td></td>
</tr>
<tr>
<td>M5 : ( E_T^{\text{miss}} = [500 - 600] \text{ GeV} )</td>
<td>53</td>
<td>87 %</td>
<td>57</td>
<td>86 %</td>
<td></td>
</tr>
<tr>
<td>M6 : ( E_T^{\text{miss}} = [600 - 700] \text{ GeV} )</td>
<td>34</td>
<td>91 %</td>
<td>36</td>
<td>91 %</td>
<td></td>
</tr>
<tr>
<td>M7 : ( E_T^{\text{miss}} &gt; 700 \text{ GeV} )</td>
<td>49</td>
<td>89 %</td>
<td>46</td>
<td>90 %</td>
<td></td>
</tr>
</tbody>
</table>
In order to test a new physics signal (corresponding to a Monte Carlo sample mysample.hep.gz or mysample.hepmc.gz), it is sufficient to start MadAnalysis 5 in the reco mode (../bin/ma5 -R) and type:

```
set main.recast = on
set main.recast.store_root = False
import <path-to-the-event-sample>
submit
```

The interpreter then asks to edit the recasting card where one can tick/untick the analyses that need to be recasted. One can also use the command:

```
set main.recast.card_path = <path-to-a-recasting-card>
```

to provide the recasting card directly. The output directory contains the events of the store_root option has been set to True, as well as the CLs an efficiency information for each signal region of each of the selected analyses. The cross section excluded at the 95% CL by each region is also provided as additional information.
Implementing Cutflows in MA5 is easy