LabNotebook

Austin Vihncent Skeeters

August 17, 2015

Contents

Part I

HSCP Analyais

Chapter 1

File Generation

1.1 Useful Twikis

- https://twiki.cern.ch/twiki/bin/viewauth/CMS/Hscp2014Analysis#Instructions_to_produce_HSCP_sam

- https://twiki.cern.ch/twiki/bin/view/Main/BatchJobs

- https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideCmsDriver

1.2 AOD Files

1. Configuration File Creation

   (a) Available MC Particle Data The directory that contains the list of PYTHIA information for the various available particles is:

   ```
   /afs/cern.ch/work/a/askeeter/private/CMSSW_7_4_4_patch4
       /src/Configuration/GenProduction/python/ThirteenTeV/
   ```

   The list of available files is:

Filename
HSCPmchamp12$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp12$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp15$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp15$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp18$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp18$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp1$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp1$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp24$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp24$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp2$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp2$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp30$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp30$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp36$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp36$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M1000TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M100TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M1400TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M1800TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M200TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M2200TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M2600TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M400TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M600TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M800TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp3$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp48$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp48$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp60$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp60$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M1000TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M100TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M1400TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M1800TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M200TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M2200TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M2600TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M400TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M600TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M800TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp6$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp9$_{\mathrm{M300TuneZ2star13TeVpythia6cff}}$.py
HSCPmchamp9$_{\mathrm{M900TuneZ2star13TeVpythia6cff}}$.py

(b) Creating by Hand

AOD files include all of the information from the simulation. That is, everything from the detector is included. We do not need all of this information, and it will get stripped down later. An example command to create configuration files for the creation of the AOD files is as follows:

```
cmsDriver.py Configuration/GenProduction/python/ThirteenTeV/HSCPmchamp6_M_1400_T
```

(c) Creating by Automation

Automation of the above has been accomplished by means of a bash script. This script will create the corresponding configuration file for the given charge, mass and event number as specified in the appropriate org code snippet variables available below in the runner.

```
#!/bin/bash
CHARGE=$1
MASS=$2
EVENTS=$3
cmsDriver.py Configuration/GenProduction/python/
    ThirteenTeV/HSCPmchamp${CHARGE}_M_${MASS}
    _TuneZ2star_13TeV_pythia6_cff.py --fileout file:
    mchamp${CHARGE}_M_${MASS}_AOD.root --mc --
    eventcontent AODSIM --datatier GEN-SIM-DIGI-AOD --
    conditions MCRUN2_74_V8 --step GEN,SIM,DIGI,L1,
    DIGI2RAW,HLT:GRun,RAW2DIGI,L1Reco,RECO --
    python_filename mchamp${CHARGE}_M_${MASS}_cfg.py
    --magField 38T_PostLS1 --geometry Extended2015 --
    customise SimG4Core/CustomPhysics/
    Exotica_HSCP_SIM_cfi.customise,
    SLHCUpgradeSimulations/Configuration/
    postLS1Customs.customisePostLS1 --no_exec -n ${
    EVENTS}
```

The above can be run in the appropriate directory by executing the following within the Org file:

```
sh cfgCreator.sh ${charge} ${mass} ${events}
```

2. Running Configuration Files

Once all of these steps, whether by hand or by automation have been completed, the user should be left with all of the requested AOD Root files, as well as the left over configuration files for everything that has been ran.

(a) Running by Hand

Now, once the appropriate configuration files are created, they need to be sent to the CERN Batch service to run. The above cmsDriver command includes all steps including the full simulation. To send files to the batch service, a script must be used. An example script is as follows:

```
#! /bin/sh
CMSSW_PROJECT_SRC="/afs/cern.ch/work/a/askeeter/
    private/CMSSW_7_4_4_patch4/src/"
CFG_FILE="mchamp6_M_1400_cfg.py"
OUTPUT_FILE="mchamp6_M_1400.root"
TOP="$PWD"

cd $CMSSW_PROJECT_SRC
eval `scramv1 runtime -sh`
cd $TOP
cmsRun $CMSSW_PROJECT_SRC/$CFG_FILE
rfcp $OUTPUT_FILE $CMSSW_PROJECT_SRC$OUTPUT_FILE
```

After the script is created though, make sure to change the file permissions with:

```
chmod 744 lxplusbatchscript.sh
```

Now the job(s) must be submitted to the batch service:

```
bsub -R "pool>30000" -q 1nw -J job1 <
    lxplusbatchscript.csh
```

Where the following options are true:

- "-R" "pool>30000" means that you want a minimum free space of 30GB

to run your job.

- "-q" 1nw means you are submitting to the 1-week que. Other available

queues are:

- 8nm (8 minutes)
- 1nh (1 hour)
- 8nh
- 1nd (1 day)
- 2nd (2 days)
- 1nw (1 week)

- 2nw
- -J job1 sets job1 as your job name
- < lxplusbatchscript.sh gives your script to the job.

Check your job status with: "bjobs" Kill jobs with "bkill -J job1" Using bkill without any job specified will kill ALL of your jobs.

(b) Running by Automation

All of the aforemntioned can be accomplished automatically by means of the following scripts:

```
#!/bin/bash

#Populate an array of all of the configuration files
shopt -s nullglob
filearray=( "HSCP_MC_cfg_Files"/* )
shopt -u nullglob

#Create a bash file for each config file
for file in "${filearray[@]}"
do
    parts=(${file//_/ })
    charge=${parts[3]}
    #Extract the number from the charge
    chargeFixed=$(echo $charge | tr -dc '0-9')
    mass=${parts[5]}

    #All of the important data has been stripped from
        the config filename
    #Now to create the bath scripts
    filename="mchamp${chargeFixed}_M_${mass}.sh"
    cfgfile="mchamp${chargeFixed}_M_${mass}_cfg.py"
    rootfile="mchamp${chargeFixed}_M_${mass}_AOD.root
        "
    #Create an empty file to be filled
    touch $filename
    #Use echo to populate the file contents. Not the
        cleanest way, but it works for a file this
        short.
    echo "#!/bin/sh">$filename
    echo 'CMSSW_PROJECT_SRC="/afs/cern.ch/work/a/
        askeeter/private/CMSSW_7_4_4_patch4/src/"'>>
        $filename
    echo """CFG_FILE='/afs/cern.ch/work/a/askeeter/
        private/CMSSW_7_4_4_patch4/src/
        HSCP_MC_cfg_Files/${cfgfile}'"""">>$filename
    echo """OUTPUT_FILE='/afs/cern.ch/work/a/askeeter
        /private/CMSSW_7_4_4_patch4/src/
        HSCP_MC_Root_Files/$rootfile'"""">>$filename
```

```
    echo "OUT_FILE='$rootfile'">>$filename
    echo 'TOP="$PWD"'>>$filename
    echo 'cd $CMSSW_PROJECT_SRC'>>$filename
    echo 'eval `scramv1 runtime -sh`'>>$filename
    echo 'cd $TOP'>>$filename
    echo 'cmsRun $CFG_FILE'>>$filename
    echo 'rfcp $OUT_FILE $OUTPUT_FILE'>>$filename
    #DO NOT FORGET to change the config file
        permissions if you are creating these by hand.
    chmod 744 $filename
done
```

The above places all created files in certain directories that are expected to remain constant. Should these change, all of the automation scripts will need to be updated, although the "by hand" methods will remain independent.

After running the batch-creator script, we will have all of the necessary batch files that we need to run in order to utilize the appropriate configuration file to produce a corresponding AOD Root file.

```
#!/bin/bash

#Populate an array of all of the batch scripts
shopt -s nullglob
filearray=( "HSCP_MC_sh_Files"/* )
shopt -u nullglob

for file in "${filearray[@]}"
do
    #Strip off the chracters that we dont need
    fileFixed=${file:17}
    #Send to the two day queue. This can be changed
    bsub -R "pool>20000" -q 2nd -J $fileFixed < /afs/
        cern.ch/work/a/askeeter/private/
        CMSSW_7_4_4_patch4/src/HSCP_MC_sh_Files/
        $fileFixed
done
```

1.3 EDM Files

1. Creation by Hand

   Once the creation of the AOD files is complete, they need to be converted into something that is a bit smaller, containing only the information that we need. Basically, this process involves cutting out some

of the "meat" of the AOD files, reducing their size, but not (at least for our purposes) their utility.

A single file needs to be modified that dictates to cmsRun which AOD file that you would like to convert to EDM. The file resembles the following:

```
import sys, os
import FWCore.ParameterSet.Config as cms
#Makes EDM from AOD
isSignal = True
isBckg = False
isData = False
isSkimmedSample = False
GTAG = 'MCRUN2_74_V8'
OUTPUTFILE = '/afs/cern.ch/work/a/askeeter/private/
    CMSSW_7_4_4_patch4/src/HSCP_MC_Root_Files/
    mchamp3_M_400_EDM.root'

#InputFileList = cms.untracked.vstring()

#debug input files
#this list is overwritten by CRAB
InputFileList = cms.untracked.vstring(
    #The comment is an example of how to do this from a
        remote directory
    #'root://cmseos.fnal.gov//eos/uscms/store/user/
        aackert/HSCP/AODgen/condorjdls/step2_condortest.
        root',
    #Below is the file that you want to conver from AOD
        to EDM
     'file:/afs/cern.ch/work/a/askeeter/private/
        CMSSW_7_4_4_patch4/src/HSCP_MC_Root_Files/
        mchamp3_M_400_AOD.root'
)


#main EDM tuple cfg that depends on the above parameters
execfile( os.path.expandvars('${CMSSW_BASE}/src/
    SUSYBSMAnalysis/HSCP/test/MakeEDMtuples/
    HSCParticleProducer_cfg.py') )
```

This file is located at: `/afs/cern.ch/work/a/askeeter/private/CMSSW_7_4_4_patch4/src/SUSYBSMA`

The lines that need to be altered here are "OUTPUTFILE", and 'file:/afs/cern/ch/...' which are basically telling the program the name of the EDM file that you would like created when the cmsRun has been ran, and the name of the input file(s). If you have had to split up a

job into multiple smaller files (that is, you have split up a large AOD into several smaller ones), simply include the names of each of those files in the "InputFileList", separated by commas and endlines. It is simply a python array.

Once this file has been altered, the conversion is accomplished simply by running the following:

```
cmsRun HSCParticleProducer_Signal_cfg.py
```

These jobs can also be sent to the batch service if you would like, however scripts would still need to be created just as with the AOD files.

2. Creation by Automation

   All of the aforementioned can be accomplished by means of two automation scripts. One script, as with the batchCreator script, is only responsible for the creation of the Python configuration file dictating the terms of the AOD to EDM conversion. The other, as is similar again, is responsible for executing the conversion jobs.

```
#!/bin/bash

#Populate an array of all of the AOD files
shopt -s nullglob
filearray=( "HSCP_MC_Root_Files"/*AOD* )
shopt -u nullglob

for file in "${filearray[@]}"
do
    parts=(${file//_/ })
    charge=${parts[3]}
    #Extract the number from the charge
    chargeFixed=$(echo $charge | tr -dc '0-9')
    mass=${parts[5]}
    aod_file="mchamp${chargeFixed}_M_${mass}_AOD.root"
    root_file="mchamp${chargeFixed}_M_${mass}_EDM.root"
    python_file="mchamp${chargeFixed}_M_${mass}_cfg.py"

    cat > /afs/cern.ch/work/a/askeeter/private/
        CMSSW_7_4_4_patch4/src/
        HSCP_MC_AODtoEDM_Python_Files/${python_file} <<
        EOF
import sys, os
import FWCore.ParameterSet.Config as cms
#Makes EDM from AOD
isSignal = True
```

```
isBckg = False
isData = False
isSkimmedSample = False
GTAG = 'MCRUN2_74_V8'
OUTPUTFILE = '/afs/cern.ch/work/a/askeeter/private/
    CMSSW_7_4_4_patch4/src/HSCP_MC_Root_Files/${root_file
    }'

InputFileList = cms.untracked.vstring(
'file:/afs/cern.ch/work/a/askeeter/private/
    CMSSW_7_4_4_patch4/src/HSCP_MC_Root_Files/${aod_file}'
)

execfile( '${CMSSW_BASE}/src/SUSYBSMAnalysis/HSCP/test/
    MakeEDMtuples/HSCParticleProducer_cfg.py' )
EOF

done
```

```
#!/bin/bash

shopt -s nullglob
filearray=( "HSCP_MC_Root_Files"/*AOD* )
shopt -u nullglob

for file in "${filearray[@]}"
do
    parts=(${file//_/ })
    charge=${parts[3]}
    #Extract the number from the charge
    chargeFixed=$(echo $charge | tr -dc '0-9')
    mass=${parts[5]}
    aod_file="mchamp${chargeFixed}_M_${mass}_AOD.root"
    root_file="mchamp${chargeFixed}_M_${mass}_EDM.root"
    python_file="mchamp${chargeFixed}_M_${mass}_cfg.py"

    #Replace the standard configuration file with the one
        currently being ran
    cp /afs/cern.ch/work/a/askeeter/private/
        CMSSW_7_4_4_patch4/src/
        HSCP_MC_AODtoEDM_Python_Files/${python_file} /afs/
        cern.ch/work/a/askeeter/private/CMSSW_7_4_4_patch4
        /src/SUSYBSMAnalysis/HSCP/test/MakeEDMtuples/
        HSCParticleProducer_Signal_cfg.py
    cd /afs/cern.ch/work/a/askeeter/private/
        CMSSW_7_4_4_patch4/src/SUSYBSMAnalysis/HSCP/test/
        MakeEDMtuples/
    cmsRun HSCParticleProducer_Signal_cfg.py
    #Don't do the next file until the previous one has
```

```
        ran to completion, as these jobs are not sent to
        batch
    wait
done
```

1.4 Usable NTuples

1. Creating by Hand

   Once we have EDM files, we are ready to create usable ROOT files that we can perform analysis on. In order to do so, we have to call on the Launch.py program located in:

   `/afs/cern.ch/work/a/askeeter/private/CMSSW_7_4_4_patch4/src/SUSYBSMAnalysis/H`

   Before calling this though, we must tell Launch which files that we would like to act on. This is accomplished by editing the "Analysis$_{\text{Samples}}$.txt" file which is located in the same directory as Launch. An example of this file is as follows:

```
#RELEASE, SAMPLE TYPE (0=data, 1=bckg, 2=signal, 3=signal systematic), SIGNAL
#HSCP Signal
#"CMSSW_7_4",  2, "MChamp9_13TeV_M900"                , "MChamp9_13TeV_M900"
#
#"CMSSW_7_4",  2, "MChamp6_13TeV_M900"                , "MChamp6_13TeV_M900"
"CMSSW_7_4",   2, "mchamp18_M_300"                    , "mchamp18_M_300_EDM"



#
#
#
#Background
#"CMSSW_7_4", 1, "MC_13TeV_DYToMuMu"                  , "MC_13TeV_DYToMuMu"
```

   There are several potential things that could be edited in this file, but we mainl only need to edit three. The first column corresponds to the version of CMSSW being used. The second corresponds to the sample type being read in, where keys and meanings are displayed at the top of the file. The third column is the desired name of the output data file once conversion is complete. The fourth column corresponds to the name of the input EDM file, with no ".root" extension. The next

column is a label to be used in plots produced by steps two and higher. The next column "s10" does not need to be changed.This is the type of pileup distribution. The next column must be changed to equal the mass of the desired particle in GeV/c^2. None of the other numbers need to be changed. It is of course possible to process multiple files in this step.

In order to produce usable tuples, we must simply run step one. Step one converts our EDM files to a usable root file. Now that the above file has been properly edited, we simply run:

```
Launch.py 1
```

Which will run step 1 of the analysis code. Upon completion, the data root tuples will be stored in the "Results" folder of the same directory as Launch. It should be noted that the jobs are auto-batched to Cern's 2 day queue.

Once can also run the jobs locally by looking in the "FARM/inputs" folder. You will see ####$_{\text{HscpAnalysis}}$.sh where the numbers correspond to the batched job number. Less them and the bottom lines will tell you what samples they are running on if you're not sure. Then just do:

```
source filename.sh >& output.txt &
```

To run locally (and redirect the output). Running locally is usually faster than sending to batch, but if the local running takes longer than two hours, the job will be killed automatically.

2. Creating by Automation

The following script will simply populate the "AnalysisSamples$_{\text{Gloabal}}$.txt" file with all of the EDM files that are available for conversion. Careful with this program though, as it merely appends to the already existing file. It is best to delete the contents of the already existing file before running this in order to avoid the creation of duplicate files.

```
#!/bin/bash

shopt -s nullglob
filearray=( "HSCP_MC_Root_Files"/*EDM* )
shopt -u nullglob
#printf "%s\n" "${filearray[@]}"
```

```
appendTo="/afs/cern.ch/work/a/askeeter/private/
    CMSSW_7_4_4_patch4/src/SUSYBSMAnalysis/HSCP/test/
    AnalysisCode/Analysis_Samples.txt"
#Create a python file for each config file
for file in "${filearray[@]}"
do
    parts=(${file//_/ })
    charge=${parts[3]}
    #Extract the number from the charge
    chargeFixed=$(echo $charge | tr -dc '0-9')
    mass=${parts[5]}
    aod_file="mchamp${chargeFixed}_M_${mass}_AOD.root"
    edm_file="mchamp${chargeFixed}_M_${mass}_EDM"
    gen_file="mchamp${chargeFixed}_M_${mass}"
    #We need to append to the Analysis_Samples.txt file
    cat >> /afs/cern.ch/work/a/askeeter/private/
        CMSSW_7_4_4_patch4/src/SUSYBSMAnalysis/HSCP/test/
        AnalysisCode/Analysis_Samples.txt << EOF
"CMSSW_7_4",    2, "$gen_file"    ,"$edm_file"    , "MC:
    mchamp${chargeFixed} ${mass} GeV/#font[12]{c}^{2}" , "
    S10", $mass, +9.8480000000E+01, 0, 1.000, 1.000, 1.000
EOF
done
```

Once the list has been populated, the next step is the same as the above. Simply run:

```
Launch.py 1
```

1.5 Plots

Step 2 of the analysis code involves generating plots. In order to run this step, you must simply use the number "2" with Launch.py, similar to the previous step. However, you must make sure to edit the file "Analysis$_{\text{Global}}$.h" located in the Launch.py directory, around line 157. Make sure that the "BaseDirectory" points to where your samples being analyzed are located. Then, you can simply run step 2 AFTER step 1.

Part II

Playing around with org examples

```
import matplotlib, numpy
#matplotlib.use('Agg')
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(4,2))
x=numpy.linspace(-15,15)
plt.plot(numpy.sin(x)/x)
fig.tight_layout()
plt.savefig('/tmp/python-matplot-fig.png')
return '/tmp/python-matplot-fig.png' # return filename to org-
    mode
```

```
#include <iostream>
using namespace std;
int main(void){
  cout << "Hello world!" << endl;
  }
```

$$-\left\langle \frac{dE}{dx} \right\rangle = 4\pi N_A r_e^2 m_e c^2 z^2 \frac{Z}{A} \frac{1}{\beta^2} \left[\frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 T_{max}}{I^2} - \beta^2 - \frac{\delta(\beta\gamma)}{2} \right] \times Q^2$$

Part III

Code Snippets

Chapter 2

Bash script populate array of all files

To populate an array of all of the files in a certain folder you can do something similar to:

```
shopt -s nullglob
filearray=( "HSCP_MC_sh_Files"/* )
shopt -u nullglob
#Now to loop through them
for file in "${filearray[@]}"
do
    #strip off the characters that we don't need
    fileFixed=${file:17}
done
```

This is what I use in my script that sends jobs to the cern batch service, as well as my script that creates the batch shell files based on the available configuration files.

Chapter 3

Bash script extract numbers from filename

Similar to the above, we must first obtain a file name that we wish to parse. So:

```
shopt -s nullglob
filearray=( "HSCP_MC_cfg_Files"/* )
shopt -u nullglob
#Now to loop through them
for file in "${filearray[@]}"
do
    parts=(${file//_/ })
    charge=${parts[3]}
    #extract the number from the charge
    chargeFixed=$(echo $charge | tr -dc '0-9')
    mass=${parts[5]}
    #Now we have our relevant info from the file!
done
```