How to create nTuples with the "Brunel nTupliser" Documentation Run 2

Setup:

Before using the Brunel nTupliser, the following CMSSW dependencies must be downloaded for CMSSW.

Electron Identification MVA:

Download the package through the following command (in the CMSSW src directory):

"*>>> git git cms-merge-topic ikrav:egm_id_7.4.12_v1*"

Following this, navigate back to the src/ directory of CMSSW. The Brunel nTupliser is then downloaded from github through the following command:

"*>>> git clone –b <CMSSW_branch> git@github.com:davidcarbonis/NTupliser.git*"

Currently the branches available are "CMSSW_5_3_20" for Run 1 data/MC and "CMSSW_7_4_7" for Run 2 data. This document concerns using the nTupliser for Run 2 only.

Part 1b - Users using Crab 3:

Within the *test/* subdirectory there are configuration files called Crab3Config_MC.py and Crab3Config_data.py which contain the necessary scripting to run the nTupilser code in *test/ nTupliserMC_cfg _cfg.py* or *test/ nTupliserData.py* respectively. The dataset (listed under the parameter `config.Data.inputDataset`), the local working directory (`config.General.requestName`), and the output location on the Brunel Tier 2 storage element (`config.Data.outLFNDirBase`) will have to be set by the user.

It goes without saying that if the user wishes to do something different to the norm in the nTupilisation process, they will need to modify *nTupliserData_cfg.py/ nTupliserMC_cfg.py* before running Crab 3.

Once the crab configuration file is set up, the usual formulas before running CMS Software will need to be cast – ie. executing "*>>> cmsenv*". The user will need to have a valid CMS VO certificate (this can be checked by excuting "*>>> voms-proxy-init -voms cms*").

To submit the job to the grid, run Crab 3. A short overview is given below, but if you encounter any difficulty, the relevant documentation for using Crab 3 can be found on the CMS Twiki.

i. Execute "*>>> crab submit -c <CRAB-config-file>*" to create and submit the jobs.
ii. Execute "*>>> crab status -d <dir-name>*" to check the progress of the submitted jobs. One can get a more detailed status report by added "—long" to the crab status command.
iii. Once all the jobs have completed execute "*>>> crab getoutput -d <dir-name>*" to retrieve the output of the jobs (just the log files as the output files are copied over to the storage element associated to the T2 specified).

Once the output logs have been retrieved one needs to create a list of all the files produced from the Crab jobs. For Run 1 there was a script to produce a list of the T2 locations of the output produced using Crab 2. Given that Crab 3 is used now, and there is not an easy method to access the physical file names (PFNs) and their locations, currently one has to retrieve the files from the T2 prior to skimming. A solution to avoid having to copy large files over to the user's area and skim the files directly from their T2 storage location is being looked into.

In the directory "*/skimMacros/*", one finds the small program used to produce the skims used in the analysis code. After compiling the exe, source the setup file and run cmsenv (as it is implicit, I'll state it explicitly, this program needs to be located in a CMSSW setup) - in that order! Afterwards, execute "*skims.exe*":

```
>>> make
```

```
>>> source setup.sh
>>> cmsenv
>>> ./skims.exe < fileListFileOutputName> <outputDataSetName>
```

N.B.

All “>>>” are referring to the start of a new line on the console – in case this wasn’t implicit enough.

Crab 3 and the skimming program have been run and tested only under CMSSW_7_4_7 by the author.

The author takes no responsibility for any unintended consequences stemming from use of these instructions. They are meant as a guide only, to provide a starting point and to give others documentation which he desperately lacked during his student days.