Automated collider event selection, plotting, & machine learning with AEACuS, RHADAManTHUS, & MInOS

Joel W. Walker[a,*]

[a] *Department of Physics and Astronomy, Sam Houston State University, Box 2267, Huntsville, TX 77341, USA*

E-mail: jwalker@shsu.edu

A trio of automated collider event analysis tools are described and demonstrated, in the form of a quick-start tutorial. AEACuS interfaces with the standard MadGraph/MadEvent, Pythia, and Delphes simulation chain, via the Root file output. An extensive algorithm library facilitates the computation of standard collider event variables and the transformation of object groups (including jet clustering and substructure analysis). Arbitrary user-defined variables and external function calls are also supported. An efficient mechanism is provided for sorting events into channels with distinct features. RHADAManTHUS generates publication-quality one- and two-dimensional histograms from event statistics computed by AEACuS, calling MatPlotLib on the back end. Large batches of simulation (representing either distinct final states and/or oversampling of a common phase space) are merged internally, and per-event weights are handled consistently throughout. Arbitrary bin-wise functional transformations are readily specified, e.g. for visualizing signal-to-background significance as a function of cut threshold. MInOS implements machine learning on computed event statistics with XGBoost. Ensemble training against distinct background components may be combined to generate composite classifications with enhanced discrimination. ROC curves, as well as score distribution, feature importance, and significance plots are generated on the fly. Each of these tools is controlled via instructions supplied in a reusable cardfile, employing a simple, compact, and powerful meta-language syntax.

Computational Tools for High Energy Physics and Cosmology (CompTools2021)
22-26 November 2021
Institut de Physique des 2 Infinis (IP2I), Lyon, France

*Speaker

1. Introduction

This article provides an introduction to three related programs, AEACuS, RHADAManTHUS, and MInOS, which respectively automate event selection, plotting, and machine learning on simulated collider data. It closely mirrors the content of a live software tutorial [1] presented by the author at the 2021 Computational Tools for High Energy Physics and Cosmology [2] workshop, hosted by the IP2I in Lyon, France. Accordingly, it is not intended to provide comprehensive documentation of the associated tools, but rather a quick-start interactive experience in the "learning by doing" style. An archive `demo.zip` containing all referenced data sets, cardfiles, and software tools (updated slightly from CompTools2021) is available for download from the workshop's Indico [1] page. While working through this tutorial, one may wish to follow along with a video recording of the live presentation, which is likewise available from the same location. Updates to the core software are distributed at GitHub [3], under the GNU General Public License [4], v3.

The operation of each software tool is specified via instructions in a reusable cardfile, employing a simple, compact, and powerful meta-language syntax. In keeping, this tutorial will focus largely on deconstructing the effect of commands stipulated in various exemplar cards. At the end of this tutorial, the reader who has followed along on their own computer should have developed a clear sense of how each software tool operates, along with a broad conception of what features and functional capabilities are available.

We will proceed in Section 2 by demonstrating how AEACuS interfaces with standard collider simulation utilities. In Section 3, we introduce the AEACuS meta-language. Section 4 explores an example physics analysis, using AEACuS to generate a lightweight event summary with computation of requested observables from raw simulation data. We next transition to demonstrating analysis tools in the context of a larger pre-generated data set (available in `demo.zip`), highlighting channel sorting with AEACuS in Section 5, then plotting with RHADAManTHUS in Section 6, and finally machine learning with MInOS in Section 7. We close with conclusions and acknowledgments.

2. Interfacing AEACuS with simulation tools

AEACuS [3] automates the generalized analysis of simulated collider events. It contains an extensive algorithm library that facilitates the computation of standard collider event observables and the classification of object groups, including jet clustering and substructure analysis. Arbitrary user-defined functions and external processing calls are also supported. An efficient mechanism is provided for sorting events into channels with distinct features. It is specifically designed to interface with the standard MadGraph/MadEvent [5], Pythia [6], and Delphes [7] simulation chain. In particular, the current distribution of AEACuS 4 [3] (beta release) has been thoroughly tested against MadGraph5_aMC@NLO v3.3.1, Pythia v8.306, and Delphes v3.5.0, with Pythia and Delphes installed using the `./bin/mg5_aMC` interface. Delphes also requires prior installation of Root [8, 9], and AEACuS derives most of its event input from the Delphes Root file output. Tests have been conducted with Root v6.24/06, installed from source with the cmake flag `-DPYTHON_EXECUTABLE=$( which python3 )`. This tutorial assumes a sufficiently similarly configured installation, running on a Unix/Linux flavor variant (including Mac OS).

We will begin with a full-stack simulation of $pp \to WZ$ events, to be processed with the instructions in Card A, which attempts to replicate the CMS Standard Model (SM) study in Ref. [10]. One should navigate via the command line to their MadGraph installation folder, and call the executable `./bin/mg5_aMC` to start an interactive session. In the MadGraph shell, define a multi-particle incorporating the two charged W-bosons with the instruction `define w = w+ w-`. Then, generate the process of interest with `generate p p > w z`. Next, stipulate the inclusive simulation of up to one additional light partonic jet, as `add process p p > w z j`. Finally, output a folder to hold the requested process, as `output WZJ`, and `exit`.

Entering the created folder `cd WZJ`, we will then navigate `cd Cards` to the card folder to make some edits. To ensure that Pythia and Delphes are called, we will copy the default cards for those tools into their active forms, as `cp pythia8_card_default.dat pythia8_card.dat` and `cp delphes_card_default.dat delphes_card.dat`. A few modifications to the `run_card.dat` should also be made, using a text editor. In particular, it is important to set the run tag, e.g. `WZJ = run_tag`, as this identifier will be carried through the full analysis. We may also wish to trim down the event production, e.g. to `2500 = nevents`, for the sake of efficiency during the demonstration. Note that it is no longer necessary to stipulate `False = use_systematics` as suggested in the video, if using a current AEACuS distribution. A copy of the analysis instructions in Card A (distributed as `cut_card_WZ.dat` in the `Cards` folder of the `demo.zip` archive from Indico [1]) should also be placed into the local `Cards` directory of the MadGraph process folder. Subsequently, we can back out `cd ../` one level of depth and initiate event generation `./bin/generate_events -f`. We suggest resubmitting this command once the first process completes, in order to demonstrate how AEACuS handles oversampling of a repeated phase space, as indicated by duplicative run tags.

The AEACuS "executable" is delivered as a single Perl script named `aeacus.pl` within the source archive at GitHub [3]. Since Perl is an interpreted language, this file serves both as the program source document and runtime portal. Benefits of this paradigm include transparency, self-sufficiency, and inherent platform independence — AEACuS is ready for immediate use, without the need for installation or compilation, on any computer with a reasonably modern Perl environment (versions `5.8.9` and following). A copy of the script `aeacus.pl` should be placed in the local binary path `./bin/internal/` and called `./bin/internal/aeacus.pl cut_card_WZ.dat` with the intended control card (defaulting to `cut_card.dat`) as an input parameter.

This invocation sets a number of processes in motion. First, Delphes Root files under the `./Events` path are located, converted to the LHCO [11] format (with extensions for the handling of per-event weights and specified auxiliary data), and stored in the `./Events` directory. Requested event observables are computed, specified cuts are applied, and results are output with statistics summarizing the selection flow as `.cut` files in the `./Cuts` directory.

3. The AEACuS meta-language

This section describes generic features of the meta-language syntax, which apply to control cards for all three programs. Perhaps the most distinctive feature of this language is the fact that it does not provide variables in the traditional sense. We will describe the structures which it employs instead as "shelves". The important concept is that a shelf exists independently of whether any goods are currently stored on it. Additionally, a shelf may be labeled with its specified function.

Each separate type of shelf is assigned a unique three-character alpha-numeric (with leading alpha) key. For example, `JET` shelves successively refine and gather groups of jet objects, `PTM` shelves extract and store transverse momentum magnitudes, and `MT2` shelves calculate and retain values of the M_{T2} statistic. Since multiple instances of a given type may be required, e.g. for housing distinct lepton classifications, shelves also carry a three-digit index, from `000` to `999`, joined to the key by an underscore, like "`LEP_001`".

Each instruction line in an AEACuS cardfile begins with a shelf identifier, e.g. `JET_001`, followed by an equals sign (`=`), and then a parameter specification consisting of key and value pairs, individually joined by a colon (`:`), and separated from adjacent pairs by a comma (`,`). Each parameter key is a three-character alphanumeric (leading alpha) string, e.g. `PRM` for pseudo-rapidity magnitude, that uniquely specifies the role of the subsequently input value. The value assigned to each key may be a number, i.e. integers or floating point numbers (including signed values and values in scientific notation like `+9.119E+01`), strings enclosed in double quotes (e.g. `"$M_Z$"`), another key, or a full shelf identifier. There is also a special syntax for function definitions, to be described subsequently. Following MadGraph, the base unit of energy and momentum in AEACuS is the GeV.

Alternatively, a list of values may be specified, containing elements drawn from the prior classes, individually separated by commas (`,`) and enclosed in square brackets, e.g `[0,2.5]`. A frequently employed formatting idiom is the "`[MIN,MAX]`" pair, consisting of a two-element list indicating a numeric range. So long as the provided values are sequentially ordered, the set of matched values are all numbers inclusively bounded by the specified range. An undefined value for `MIN` is treated as indefinitely small, and an undefined value for `MAX` is treated as indefinitely large. If no numerically valid limits are provided, then all values match. If the upper and lower boundaries are numerically equivalent, then only that single value matches. However, a subtle additional functionality is accessed under the circumstance that provided values are out of numerical sequence. In this case, a match is achieved if the comparison value is either at least as large as `MIN` or at least as small as `MAX`; in other words, the interval from `MAX` up to `MIN` is rejected, exclusive of the boundaries.

Shelf designations must be placed at the beginning of a line, with no leading whitespace, and continuation from the prior line is indicated through simple indentation, with any number of spaces or tabs. Comments are initiated with the hash/pound (`#`) character, extending to the end of the line. The order in which lines are specified in the cardfile is not material to the sequence in which they are ultimately evaluated. Rather, program execution follows a prescribed order through shelf keys in conjunction with an ascending numerical sort on shelf indices.

4. Event selection with AEACuS

This section focuses on interpreting instructions from the example analysis in Card A. Starting on Line 3, all retained electron candidates are required to have a transverse momentum magnitude $P_T > 7$ GeV and a pseudorapidity magnitude $|\eta| < 2.5$. The cut applied to this line requires at least 0 such leptons. This is not therefore a meaningful cut, but its inclusion will force this shelf to be represented in the output `.cut` file. Next, candidate muons and jets are similarly restricted.

```
 1  # 1901.03428 WZ Analysis

 2  # Object Reconstruction

 3  ELE_000 = PRM:[0,2.5], PTM:7, CUT:0
 4  MUO_000 = PRM:[0,2.4], PTM:5
 5  JET_000 = PTM:30, PRM:[0,4.5]

 6  JET_001 = SRC:+000, CMP:000, CDR:[0,0.4], CUT:[0,0]
 7  JET_002 = SRC:+000, HFT:1, CUT:[0,0]
 8  JET_003 = SRC:+000, SET:LED, OUT:[PTM_004]

 9  LEP_001 = SRC:+000, EMT:-3, CUT:[3,3]
10  LEP_002 = SRC:+001, SET:[DIL,-1,+1,91.2,15], PTM:10, CUT:2,
11            OUT:[PTM_001,PTM_002]
12  LEP_003 = SRC:+002, SET:LED, PTM:25, CUT:1
13  LEP_004 = SRC:[+001,-002], PTM:25, CUT:1,
14            OUT:[EP0_003,EP1_003,EP2_003,EP3_003,PTM_003]
15  LEP_005 = SRC:+002, EFF:SUM, OUT:[MAS_011,PTM_011]
16  LEP_006 = SRC:+001, EFF:SUM,
17            OUT:[EP0_010,EP1_010,EP2_010,EP3_010,MAS_010]
18  LEP_007 = SRC:+001, SET:[DIL,UNDEF,UNDEF,0,4], CUT:[0,0]
19  JET_004 = SRC:+000, LEP:000, EFF:[SUM,0,0,1],
20            OUT:[EP1_006,EP2_006]

21  # Event Selection

22  CAL_000 = OUT:[1,1,1]
23  MET_000 = CUT:30
24  OTM_001 = MET:000, LEP:001

25  VAR_001 = VAL:{ ( $3*(80.4^2+2*$1*$4+2*$2*$5) - SRT(($1^2+$2^2+$3^2) *
26            ( 80.4^4 + 4*80.4^2 * ($1*$4+$2*$5) - 4*($2*$4-$1*$5)^2 ))) / ( 2*($1^2+$2^2)),
27            EP1_003,EP2_003,EP3_003,EP1_006,EP2_006 }
28  VAR_002 = VAL:{ ( $3*(80.4^2+2*$1*$4+2*$2*$5) + SRT(($1^2+$2^2+$3^2) *
29            ( 80.4^4 + 4*80.4^2 * ($1*$4+$2*$5) - 4*($2*$4-$1*$5)^2 ))) / ( 2*($1^2+$2^2)),
30            EP1_003,EP2_003,EP3_003,EP1_006,EP2_006 }
31  VAR_003 = VAL:{ SRT( MIN(
32            ( $1 + SRT($5*$5+$6*$6+$7*$7))^2 - ($2+$5)^2 - ($3+$6)^2 - ($4+$7)^2,
33            ( $1 + SRT($5*$5+$6*$6+$8*$8))^2 - ($2+$5)^2 - ($3+$6)^2 - ($4+$8)^2)),
34            EP0_010,EP1_010,EP2_010,EP3_010,EP1_006,EP2_006,VAR_001,VAR_002 }
```

Card A: Rendering of CMS $pp \to WZ$ study from Ref. [10].

Proceeding to Line 6, we next begin to form compounded sets of jets and leptons. Whereas the zeroth member of each category is automatically supplied, subsequent entries must stipulate the source SRC category or categories from which they are drawn. JET_001 is built from the zeroth jet classification, and members must have a comparative delta-R of less than 0.4 radians from the comparison set (jets compare against leptons and leptons compare against jets), namely the zeroth lepton shelf. The event is rejected by application of a CUT if the number of members having the described characteristics is not exactly zero. In other words, this line vetoes events where a candidate jet is on top of a poorly isolated candidate lepton. Next, admission to jet classification 002 requires a heavy flavor tag, but the membership of this class must be zero. In other words, this is a b-jet veto. Classification 003 is formed from a subset of the source category using the algorithm LED, which extracts just the leading member by transverse momentum magnitude. The transverse momentum of this jet is extracted and stored on the fourth PTM shelf.

On Line 9, a similar subclassification of leptons is begun. The first lepton shelf has a flavor (electron/muon/tau) restriction of “not 3”, which amounts to exclusion of hadronic taus, and the event is vetoed unless there are precisely three electron/muon candidates. The second lepton classification takes the first classification as its source and extracts a subset of member objects using the dilepton selection algorithm. Parameters indicate that this should be an opposite-sign, same-flavor dilepton, with reconstructed mass as close as possible to the Z-boson mass, and not further from this mass than 15 GeV. Members are required to have a transverse momentum magnitude $P_T > 7$ GeV (these values are also output), and there must be two such members surviving. The third classification proceeds from the second, extracting the leading member, and increasing the applicable selection to $P_T > 25$ GeV. The fourth classification is a compound set, sourcing members from set one (the three leptons) that are not also members of set two (the reconstructed Z). In other words, it represents the lepton associated with the candidate W-boson. The transverse momentum magnitude selection is again elevated, and that value is output, along with all four-vector coordinates of the associated object. The fifth classification sources from the dilepton, and introduces a new keyword for the creation of an effective object, applying the four-vector summing algorithm, and outputting the resultant mass and transverse momentum magnitude. The sixth classification similarly sums the trilepton, and generates output. Classification seven builds a dilepton candidate from the three classified leptons, with no restrictions on sign or flavor, preferencing a reconstructed mass of zero, within 4 GeV. The applied cut then serves to veto events where all constructible dilepton pairs do not have a mass of at least 4 GeV. The fourth jet classification actually constructs a sort of missing transverse momentum vector, summing all jets together with all leptons and inverting the resulting four vector. The parameters to the `SUM` algorithm are Boolean flags for various transformations, respectively masslessness, transverseness, and momentum inversion. The resulting x and y momentum components are output.

Starting on Line 22, we transition from object reconstruction to event selection. The calorimetric missing transverse energy (more precisely, the value exported by Delphes) is output, with flags indicating that the components and their associated azimuthal angle should likewise be computed and stored. The missing transverse energy as summed internally by AEACuS is required satisfy $\not{E}_T > 30$ GeV. Likewise a transverse mass is computed from the zeroth `MET` and the trilepton system.

Line 25 and following exhibit a powerful language feature that allows users to compute new observables as arbitrary functions of previously stored values. Such functions are wrapped in curly braces `{}`, where the zeroth entry defines the functional form, and subsequent entries list the function inputs. The first input is aliased to `$1` in the formula, the second to `$2`, and etc. Referencing prior outputs, the five inputs to the first custom variable function are momentum components (P_x^W, P_y^W, P_z^W) and $(\not{P}_x, \not{P}_y)$, respectively associated with the candidate W-boson and the inverted four vector sum in the fourth jet classification. In terms of these inputs, together with the known W–boson mass, the prescribed calculation may be expressed as follows.

$$\frac{P_z^W \times \left[M_W^2 + 2P_x^W \not{P}_x + 2P_y^W \not{P}_y\right] - \sqrt{\left[(P_x^W)^2 + (P_y^W)^2 + (P_z^W)^2\right] \times \left[M_W^4 + 4M_W^2 \times (P_x^W \not{P}_x + P_y^W \not{P}_y) - 4 \times (P_y^W \not{P}_x - P_x^W \not{P}_y)^2\right]}}{2 \times \left[(P_x^W)^2 + (P_y^W)^2\right]}$$

After calling AEACuS in a simulation folder as described in the prior section, an output file is created in the `./Cuts` folder for each processed LHCO event record. This file contains headers summarizing the event selection flow, together with matrices detailing the overlap of any cuts applied in parallel. Each requested event statistic is then tabulated for every surviving event, in a manner that facilitates subsequent reprocessing by all three programs. Multiple statistically independent simulations of a common final state are named with a unique trailing index.

5. Event channel sorting with AEACuS

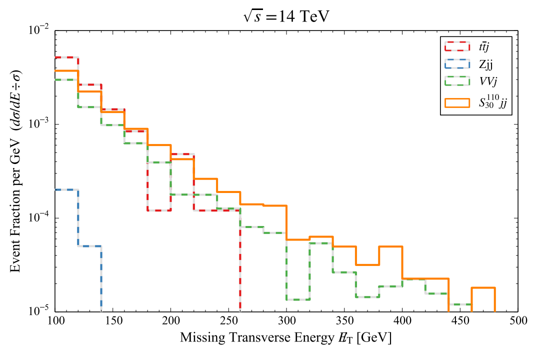
In this section and those following we will be working with a larger preprocessed data set, distributed as part of the described `demo.zip` archive. These samples were produced with Card B, as adapted from the supersymmetry search with small mass gaps in Ref. [12]. We will not elaborate the line-by-line operation of this card, but will instead focus on a few features not previously described. For reference, the various event selection shelves are storing values of the missing transverse energy, the ditau mass statistic, $\cos\theta^*$, M_{T2}, the MET delta-phi angle, various azimuthal object separations, and a collection of user-defined variables.

Line 4 of Card B introduces the concept of event selection channels. The zeroth channel encompasses all previously described functionality, and related keys may be referenced explicitly in order to customize its operation. In this case we stipulate directories (two levels up) for the storage of generated `.lhco` and `.cut` files. This is particularly useful when doing large simulations in a cluster environment. Several copies of the process subfolder may be deployed simultaneously for oversampling of the target phase space, while output files are accumulated in a common location, and automatically assigned a unique incremented index.

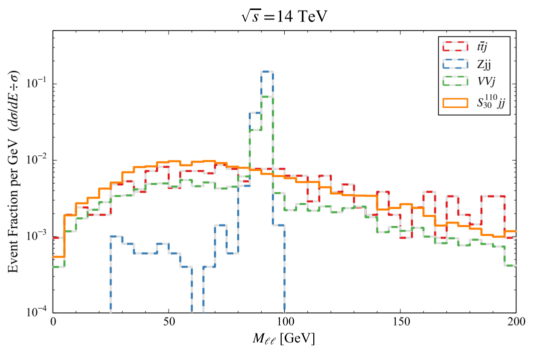
Starting in Line 48, we present an example of channel sorting. First, an event selection cut is defined (though not yet applied), based on previously computed observables. In this case case the event selection retains monojet events, according to the first jet classification. Next, positively indexed channels are defined by subscription to some number of event selections, referenced by their shelf index. Negation of the index is used for anti-selection. Presently, we are generating two event channels, the first of which contains monojet events, and the second everything else. Specifically, the second set features dijet events, as may be seen by inspecting the definition of the referenced jet classification.

The `CHN_001` monojet outputs are collected under the `./Cuts` folder of the demonstration archive, and will form the basis of all subsequent analysis. Readers are encouraged to explore that folder, where they will find records associated with a supersymmetry model featuring a 110 GeV slepton and an 80 GeV neutralino, as well as SM $t\bar{t}$, Z, ZZ, WZ, and WW final states, all with inclusive simulation of one or more additional jets. There are five batches of statistically independent simulation for the supersymmetric final state, and eighty batches (to promote sufficient residual statistical power after event selection) of the SM final states, each starting from some tens-of-thousands of events. The downstream RHADAMANTHUS and MINOS utilities treat files with distinct base run tags as non-overlapping final states, whereas repeated tags with distinct trailing indices are treated as oversampling. It is recommended that the scattering process and any tranching cuts be embedded in the selected run tag name. It is also a good habit to close with the collision energy, e.g. “`_14TeV`”, since ending on an alpha character in this manner prevents the accidental clipping of otherwise trailing digits.

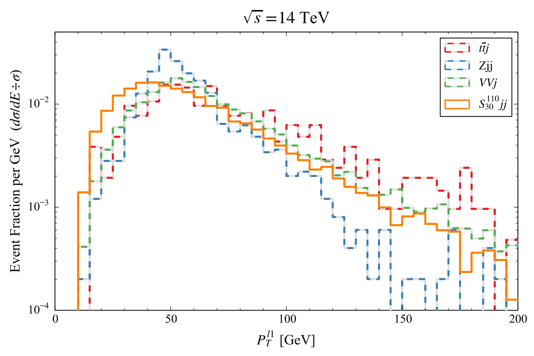
It is also possible to perform secondary channel sorting on `.cut` files after the initial processing phase. This will be exhibited presently, referencing instructions in Card C, which is also distributed as `cut_card_flow.dat` in the `Cards` folder of the demonstration archive. In this card, a number



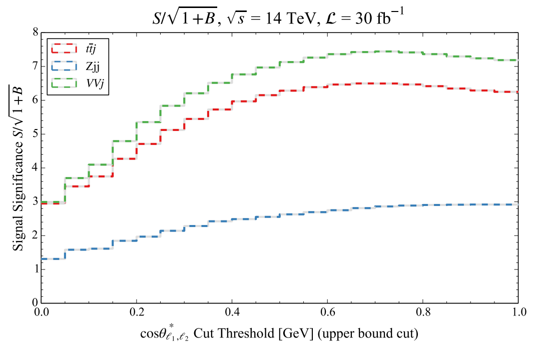
(a) Missing transverse energy shape plot.



(b) Dilepton mass shape plot.



(c) Leptonic transverse momentum shape plot.



(d) Signal significance threshold plot.

Figure 1: Histogram plotting examples.

of event selection cuts are defined, referencing observables computed with Card B. Note, as in selection `103`, that it is also possible to create an event selection `KEY` on the fly, using the function syntax. In this case, the MET delta-phi angle is simply rescaled to units of *pi*. Since we are no longer within an enclosing MadGraph process folder, it is necessary in the channel definitions to stipulate which directories and files are to be reprocessed. For the sake of example, we take all Z plus inclusive jet samples in `CHN_001` of the local `./Cuts` folder. Three channel sortings are defined by subscription to various defined selections. Readers may test the described functionality by navigating to the folder where `demo.zip` has been unpacked and issuing the command `./aeacus.pl ./Cards/cut_card_flow.dat`. Note that the card location can be anywhere on the system if specified explicitly with leading path information. This operation should result in three new channel subfolders created underneath the original source channel and containing the filtered event records.

6. Plotting with RHADAManTHUS

RHADAManTHUS generates publication-quality one- and two-dimensional histograms from event statistics computed by AEACuS, invoking MatPlotLib [13] on the back end. Large batches of simulation (representing either distinct final states and/or oversampling of a common phase space) are merged internally, and per-event weights are handled consistently throughout. Arbitrary bin-wise functional transformations are readily specified, e.g. for visualizing signal-to-background significance as a function of cut threshold.

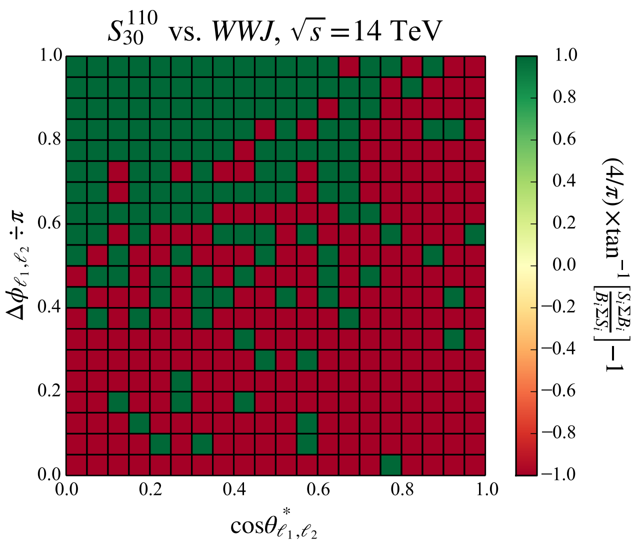


Figure 2: Two-dimensional histogram example.

Figures 1 and 2 have been generated from the `.cut` files in the demonstration archive, based on instructions contained in Card D, which is also distributed as `plt_card.dat` in the subdirectory `Cards`. The reader is invited to reproduce these plots by calling `./rhadamanthus.pl` from the folder to which `demo.zip` is unpacked. The cardfile may be omitted because it corresponds to the default name and location. This operation should result in the creation of a new folder named `Plots` containing the requested plots. If the MatPlotLib library is not installed then Python script literals will be delivered instead of figures. Execution of these scripts on a compliant system will yield the desired graphics.

We proceed now with an analysis of commands itemized in Card D. Starting on Line 1, a number of data sets are defined by referencing file names and source directories. The wildcard character `*` is used here to select multiple files, and it is also possible to provide multiple entries for `FIL`. In Line 5, data sets are assembled into channels, together with a plotting `KEY` computed previously by AEACuS.

In Line 8, the zeroth histogram key is used to store defaults. In particular, this example disables Python 3, hatching, and filling. It further enables a bolded line for the first three data sets plotted, while turning it back off for the fourth. The output format is set to `"PDF"`. Plots are normalized with a logarithmic vertical axis and no bin summing or bin smoothing. The plot title and legend (with one entry per data set) are provided as character strings, with embedded LaTeX equation formatting. A list of custom plotting colors is also provided, in the form of hexadecimal strings.

Positively indexed histograms are slated for output next, starting from Line 14. Channel shelves are referenced by index in order to indicate which observables and data sets are to be represented. A binning specification requires defined values for three keys out of the four representing left boundary, right boundary, bin span, and bin count. Minimal and maximal values may be provided explicitly for the plotting range. Horizontal and vertical axis labels are formatted similarly to the plot title and legend. Various text shortcuts are made available, e.g. `<DEF>`, which renders as the differential event fraction $d\sigma/dE \div \sigma$. Finally, a name is provided for the output plot. Normalized signal versus background shape plots such as those produced in histograms `001`, `002`, and `003` (cf. panels a, b, and c of Figure 1) facilitate the visual identification of concentrations, inflections, and knees in event distributions, informing the construction of optimized discriminants.

Various advance features are presented next, starting from the definition of a new event selection cut in Line 32. This particular selection vetoes events where the first mass shelf value lies within a 20 GeV wide window about the Z-boson. Two new channels are defined with a common key, each subscribing to the prior cut. The first channel references the three SM data sets, while the second references the signal data. Histogram `101` overrides several of the default assignments, introducing a rightward summation, turning off normalization, smoothing with a width of two bins, and disabling the log axis option. The undefined plot range values indicate that suitable boundaries should be selected automatically. Since the plot is not normalized, it is important to specify the projected luminosity, set here to 30 events per femtobarn. The plotting channel is a functional transformation of previously defined channels corresponding to the bin-wise significance ratio $S/\sqrt{1+B}$. Three such significance curves are plotted, with the single signal data set broadcast across all three data sets for the background. The requested bin summing yields an upper bound selection threshold plot in the horizontal axis observable. Panel d of Figure 1 suggests that such a cut is not immediately beneficial in the current context.

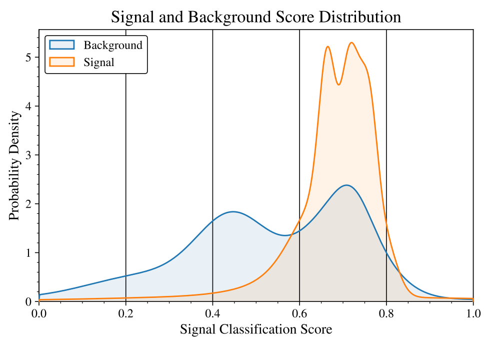
Preparation for an example two-dimensional histogram begins in Line 45. Pairs of new data sets and channels are defined, and associated with a new event selections. Two plotting keys are provided, one of which is a key function, corresponding to the two plotting axes. A two-dimensional histogram has a single plotting channel, which is assigned here to a functional transformation of the signal and background. The output in Figure 2 suggests that further event selections focus on parameter regions to the upper-left.

7. Machine learning with MInOS

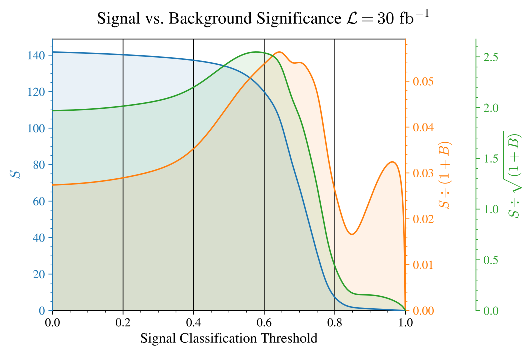
MInOS implements machine learning on computed event statistics with XGBoost [14]. Ensemble training against distinct background components may be combined to generate composite classifications with enhanced discrimination. ROC curves, as well as score distribution, feature importance, and significance plots are generated on the fly.

Figures 3 and 4 have been generated from the `.cut` files in the demonstration archive, based on instructions contained in Card E, which is also distributed as `min_card.dat` in the subdirectory `Cards`. The reader is invited to reproduce these plots by calling `./minos.pl` from the folder to which `demo.zip` is unpacked. The cardfile may be omitted because it corresponds to the default name and location. This operation should result in the creation of a new folder hierarchy `Models/TRN_010`. Subdirectories `CSV` and `Plots` contain reduced event data and plots. If the XGBoost library is not installed then a Python script literal will be delivered but not run. Execution of this script on a compliant system will yield the desired training, testing, and graphical output.

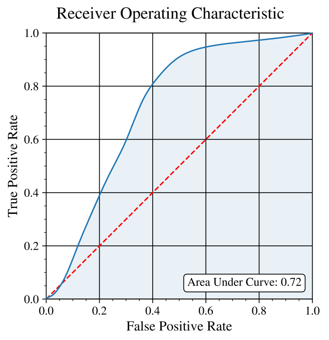
We proceed now with an analysis of commands itemized in Card E. The reader will observe several similarities to the structure of a plotting card. Data sets are defined starting on Line 1, as before. The zeroth training shelf may be used for the specification of defaults, as in Line 5. Presently, it provides the list of observable keys to be included in training and also a list of manual overrides for the LaTeX formatting of selected keys. Event selection cuts are itemized, starting on



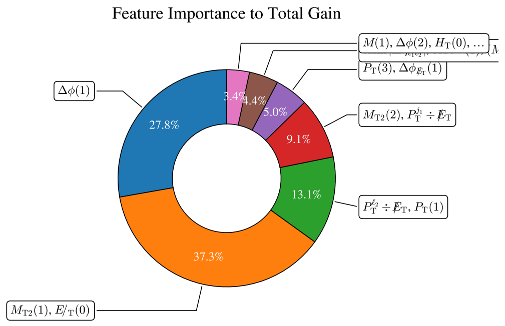
(a) Classification density.



(b) Significance curves.

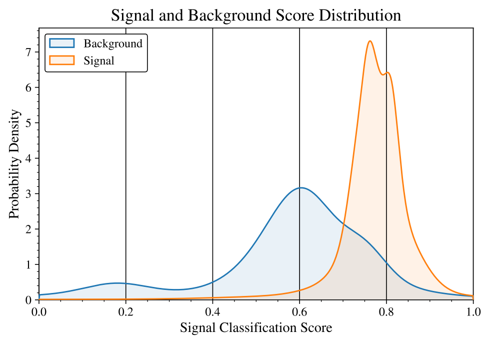


(c) ROC curve.

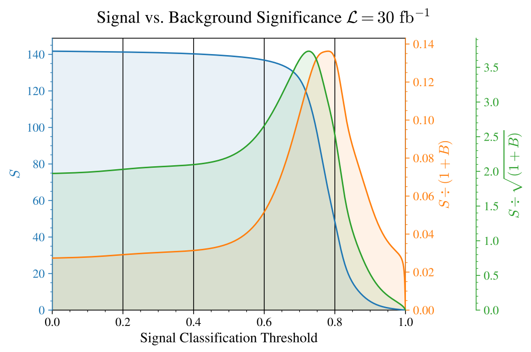


(d) Feature importance.

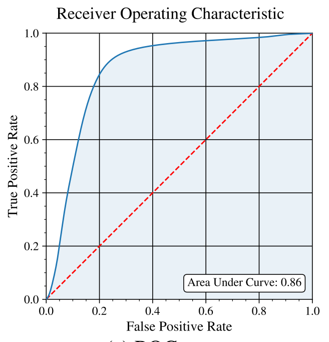
Figure 3: Joint training.



(a) Classification density.



(b) Significance curves.



(c) ROC curve.

Figure 4: Ensemble training.

Line 21. Channels are built from data sets, starting on Line 23. Channels are associated with any desired cuts and assigned a category label, e.g. 0 for background and 1 for signal. Training is requested on Line 25, referencing each of the previously defined channels.

The Boosted Decision Tree (BDT) assigns each member of the test data set a score on the continuum from 0 (background like) to 1 (signal like). Panel a of Figure 3 suggests that events known to be background (blue curve) are indeed assigned systematically lower training scores than events known to be signal (orange curve), though the two distributions are not well isolated in this case. Depending upon the placement of a threshold cut along the signal classification axis, there will be a variable instance of false positive and false negative assignments. The significance curves in panel b facilitate visual optimization of this transition point, in terms of the surviving event count S, the regulated signal to background ratio $S/(1+B)$, which is relevant to controlling systematics, and the regulated signal significance $S/\sqrt{1+B}$ at a certain target luminosity. The ROC curve in panel c provides a standardized metric of signal and background separability, where better performance is indicated by extension of the shaded blue area into the upper-left corner of the plot. BDTs are known to provide good process transparency relative to many other types of machine learning, and the feature importance chart in panel d is used to rank the discriminating power of individual observables.

The plots in Figure 3 were established by merging all events (with correct cross-section weighting) within each label category in anticipation of a single joint training. However, MInOS also generates a matrix of distinct classification scores by training separately on pairings of individual data sets comprising each label category. A composite classification score assembled from this ensemble of trainings can sometimes give better separation than a single joint training. The plots in Figure 4 are of this latter type.

8. Conclusions

The programs AEACuS, RHADAManTHUS, and MInOS have been introduced with a tutorial including execution of code on reference data and elaboration of example process control cards. These programs are maintained at GitHub [3] and are freely available for public download (GNU GPLv3[4]). The author is quite open to inquiries, suggestions, and requests for support on any topic regarding application of this software to the physics analysis of simulated collider data.

Acknowledgments

This work has been supported in part by NSF grant PHY-2112799. Thanks are extended to coauthors on a number of phenomenology studies that occasioned development and expansion of the software described in this article, including Dimitri Nanopoulos, Bhaskar Dutta, Jason Kumar, and SHSU faculty colleague William Shepherd. Several outstanding undergraduate students at SHSU must be likewise acknowledged, including Ashen Fernando, Kebur Fantahun, Nicolle Schachtner, William Ellsworth, Kamal Lamichhane, Chris Phillips, Matthew Biasucci, Tao Zhou, Alyssa Horne, and Marcus Snedeker, for participation in research projects using these tools, and for making various important contributions to their improvement. Finally, chair Nazila Mahmoudi and the

other organizers of CompTools2021 are warmly thanked for the opportunity to present a software description and quick-start tutorial.

References

[1] J.W. Walker, "Tutorial for AEACuS, RHADAManTHUS, & MInOS." `https://indico.cern.ch/event/1076291/contributions/4609972/`, 2021.

[2] N. Mahmoudi et al., "Computational Tools for High Energy Physics and Cosmology, 22–26 Nov 2021, IP2I, Lyon." `https://indico.cern.ch/event/1076291/`, 2021.

[3] J.W. Walker, "AEACuS 4.000: A Consumer-Level Tool for Implementing Generic Collider Data Selection Cuts in the Search for New Physics." `https://github.com/joelwwalker/AEACuS`, 2021.

[4] "Gnu general public license, version 3." `http://www.gnu.org/licenses/gpl.html`, 2007.

[5] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, *JHEP* **07** (2014) 079 [`1405.0301`].

[6] T. Sjöstrand, S. Ask, J.R. Christiansen, R. Corke, N. Desai, P. Ilten et al., *An introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159 [`1410.3012`].

[7] DELPHES 3 collaboration, *DELPHES 3, A modular framework for fast simulation of a generic collider experiment*, *JHEP* **02** (2014) 057 [`1307.6346`].

[8] I. Antcheva et al., *ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization*, *Comput. Phys. Commun.* **182** (2011) 1384.

[9] R. Brun and F. Rademakers, *ROOT: An object oriented data analysis framework*, *Nucl.Instrum.Meth.* **A389** (1997) 81.

[10] CMS collaboration, *Measurements of the pp $\rightarrow$ WZ inclusive and differential production cross section and constraints on charged anomalous triple gauge couplings at $\sqrt{s}$ = 13 TeV*, *JHEP* **04** (2019) 122 [`1901.03428`].

[11] "Lhc olympics wiki." `http://www.jthaler.net/olympicswiki/`, 2007.

[12] B. Dutta, K. Fantahun, A. Fernando, T. Ghosh, J. Kumar, P. Sandick et al., *Probing Squeezed Bino-Slepton Spectra with the Large Hadron Collider*, *Phys. Rev. D* **96** (2017) 075037 [`1706.05339`].

[13] J.D. Hunter, *Matplotlib: A 2d graphics environment*, *Computing in Science & Engineering* **9** (2007) 90.

[14] T. Chen and C. Guestrin, *XGBoost: A scalable tree boosting system*, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, (New York, NY, USA), pp. 785–794, ACM, 2016, DOI.

```
# 1412.0618 MT2 Han/Liu
# 1409.7058 Baer, Mustafayev, Tata
# 1706.05339 Dutta, Fantahun, Fernando, Ghosh, Kumar, Sandick, Stengel, Walker

CHN_000 = LHC:"../../LHCO", OUT:"../../Cuts"

# Object Reconstruction

ELE_000 = PRM:[0,2.5], PTM:0
MUO_000 = PRM:[0,2.5], PTM:0
TAU_000 = PRM:[0,2.5], CUT:[0,0]
JET_000 = PTM:20, PRM:[0,4.5]

JET_001 = SRC:+000, PTM:30, CUT:[1,2]
JET_002 = SRC:+001, PRM:[2.5,UNDEF], CUT:[0,0]
JET_003 = SRC:+001, SET:LED, OUT:[PTM_003,ETA_003]
JET_004 = SRC:[+001,-003], OUT:[PTM_004,ETA_004]
JET_005 = SRC:+001, EFF:SUM, OUT:MAS_003
JET_006 = SRC:+000, HFT:1, PRM:[0,2.5], CUT:[0,0]

LEP_001 = SRC:+000, EMT:+2, SET:[DIL,-1,+1,0,UNDEF], CUT:2
LEP_002 = SRC:+001, EFF:SUM, OUT:MAS_001
LEP_003 = SRC:+001, SET:LED, OUT:[PTM_001,ETA_001]
LEP_004 = SRC:[+001,-003], OUT:[PTM_002,ETA_002]

# Event Selection

MET_000 = CUT:30
TTM_001 = LEP:001, JET:005
CTS_001 = LEP:001

MT2_001 = MET:000, MOD:[GEN,LEP_003,LEP_004,100,100]
MT2_002 = MET:000, MOD:[GEN,LEP_003,LEP_004,0,0]

MDP_001 = MET:000, LEP:003
MDP_002 = MET:000, LEP:004
MDP_003 = MET:000, JET:003
MDP_004 = MET:000, JET:004

ODP_001 = LEP:001
ODP_002 = LEP:003, JET:003
ODP_003 = LEP:003, JET:004
ODP_004 = LEP:004, JET:003
ODP_005 = LEP:004, JET:004
ODP_006 = JET:001

VAR_001 = VAL:{ TNH( ABS($2-$1)), ETA_001, ETA_002 }
VAR_002 = VAL:{ TNH( ABS($2-$1)), ETA_001, ETA_003 }
VAR_003 = VAL:{ TNH( ABS($2-$1)), ETA_001, ETA_004 }
VAR_004 = VAL:{ TNH( ABS($2-$1)), ETA_002, ETA_003 }
VAR_005 = VAL:{ TNH( ABS($2-$1)), ETA_002, ETA_004 }
VAR_006 = VAL:{ TNH( ABS($2-$1)), ETA_003, ETA_004 }

VAR_011 = VAL:{ $1/$2, PTM_001, MET_000 }
VAR_012 = VAL:{ $1/$2, PTM_002, MET_000 }
VAR_013 = VAL:{ $1/$2, PTM_003, MET_000 }
VAR_014 = VAL:{ $1/$2, PTM_004, MET_000 }

VAR_021 = VAL:{ ($1-100)/$2, MT2_001, MT2_002 }

# Event Sorting

ESC_001 = KEY:JET_001, CUT:[1,1]

CHN_001 = ESC:[+001]
CHN_002 = ESC:[-001]
```

Card B: A search for supersymmetry with small mass gaps, cf. Ref. [12].

```
# Secondary channel sorting

CUT_ESC_101 = KEY:MAS_001, CUT:[96,86]
CUT_ESC_102 = KEY:TTM_001, CUT:200
CUT_ESC_103 = KEY:{$1/PI(),MDP_001}, CUT:[0,0.4]
CUT_ESC_104 = KEY:MET_000, CUT:200

CUT_ESC_201 = KEY:MAS_001, CUT:[96,86]
CUT_ESC_202 = KEY:TTM_001, CUT:400
CUT_ESC_203 = KEY:{$1/PI(),ODP_001}, CUT:0.5
CUT_ESC_204 = KEY:{$1/PI(),MDP_001}, CUT:[0,0.8]
CUT_ESC_205 = KEY:{$1-110,MT2_001}, CUT:[0,30]
CUT_ESC_206 = KEY:MET_000, CUT:400

CUT_ESC_301 = KEY:MAS_001, CUT:[96,86]
CUT_ESC_302 = KEY:TTM_001, CUT:200
CUT_ESC_303 = KEY:PTM_001, CUT:80
CUT_ESC_304 = KEY:PTM_002, CUT:80
CUT_ESC_305 = KEY:{$1/PI(),MDP_001}, CUT:[0,0.8]
CUT_ESC_306 = KEY:{$1/PI(),MDP_002}, CUT:[0,0.6]
CUT_ESC_307 = KEY:MET_000, CUT:280

CUT_CHN_011 = DIR:"./Cuts/CHN_001", FIL:"ZJ*", ESC:[+101,+102,+103,+104]

CUT_CHN_012 = DIR:"./Cuts/CHN_001", FIL:"ZJ*", ESC:[+201,+202,+203,+204,+205,+206]

CUT_CHN_013 = DIR:"./Cuts/CHN_001", FIL:"ZJ*", ESC:[+301,+302,+303,+304,+305,+306,+307]
```

Card C: Secondary channel sorting instructions

```
DAT_001 = DIR:"./Cuts/CHN_001", FIL:"TTBarJ_*"
DAT_002 = DIR:"./Cuts/CHN_001", FIL:"ZJJ_*"
DAT_003 = DIR:"./Cuts/CHN_001", FIL:["WWJ_*","WZJ_*","ZZJ_*"]
DAT_011 = DIR:"./Cuts/CHN_001", FIL:"MulMulJJ_*"

CHN_001 = DAT:[001,002,003,011], KEY:MET_000
CHN_002 = DAT:[001,002,003,011], KEY:MAS_001
CHN_003 = DAT:[001,002,003,011], KEY:PTM_001

HST_000 =
    PY3:-1, HCH:-1, FLL:-1, BLD:[+1,+1,+1,-1], FMT:"PDF",
    SUM:0, NRM:1, AVG:0, LOG:1,
    TTL:"$\sqrt{s} = 14$ TeV",
    LGD:[ "$t\bar{t}j$","Zjj","$VVj$", "$S_{30}^{110}jj$" ],
    CLR:[ "e41a1c", "377eb8", "4daf4a", "ff7f00" ]

HST_001 =
    CHN:001,
    LFT:100, RGT:500, SPN:20, BNS:UNDEF,
    MIN:.00001, MAX:.01,
    LBL:["Missing Transverse Energy <MET> [GeV]", "Event Fraction per GeV  (<DEF>)"],
    NAM:"shape_MET_n1"

HST_002 =
    CHN:002,
    LFT:0, RGT:200, SPN:5, BNS:UNDEF,
    MIN:0.0001, MAX:.5,
    LBL:["$M_{\ell \ell}$ [GeV]", "Event Fraction per GeV  (<DEF>)"],
    NAM:"shape_MAS_001_n1"

HST_003 =
    CHN:003,
    LFT:0, RGT:200, SPN:5, BNS:UNDEF,
    MIN:.0001, MAX:.05,
    LBL:["$P_{T}^{l1}$ [GeV]", "Event Fraction per GeV  (<DEF>)"],
    NAM:"shape_PTM_L1_n1"

ESC_001 = KEY:MAS_001, CUT:[101,81]
CHN_101 = DAT:[001,002,003], KEY:CTS_001, ESC:[+001]
CHN_111 = DAT:011, KEY:CTS_001, ESC:[+001]

HST_101 =
    IFB:30,
    CHN:{$2/SRT(1+$1),101,111},
    LFT:0, RGT:1, SPN:.05, BNS:UNDEF,
    MIN:UNDEF, MAX:UNDEF,
    SUM:+1, NRM:0, AVG:2, LOG:0,
    TTL:"<SR1>, $\sqrt{s}$ = 14 TeV, <LUM> = 30 <IFB>",
    LBL:[ "$\cos \theta^*_{ \ell_1,\ell_2}$ Cut Threshold [GeV] (upper bound cut)",
        "Signal Significance <SR1>" ],
    NAM:"sig_CTS_srb_ubc"

ESC_002 = KEY:MET_000, CUT:75
DAT_203 = DIR:"./Cuts/CHN_001", FIL:"WWJ_*"
DAT_211 = DIR:"./Cuts/CHN_001", FIL:"MulMulJJ_*"
CHN_201 = DAT:203, KEY:[CTS_001,{$1/PIE(),ODP_001}], ESC:[+002]
CHN_211 = DAT:211, KEY:[CTS_001,{$1/PIE(),ODP_001}], ESC:[+002]

PLT_H2D_001 =
    PY3:-1, FMT:"PDF",
    CHN:{(4/PIE()*ATN($2,$1)-1),201,211},
    LFT:0, RGT:1, BNS:20, NRM:-1, MIN:-1, MAX:+1,
    TTL:[ "$S_{30}^{110}$ vs. $WWJ$, $\sqrt{s} = 14$ TeV",
        "$(4/\pi)\times\tan^{-1}\left[\frac{S_i \Sigma B_i}{B_i \Sigma S_i}\right]-1$" ],
    LBL:[ "$\cos \theta^*_{ \ell_1,\ell_2}$", "$\Delta \phi_{\ell_1,\ell_2} \div \pi$" ],
    NAM:"H2D_CST_ODP"
```

Card D: Plotting cardfile example, cf. Ref. [12]

```
DAT_001 = DIR:"./Cuts/CHN_001", FIL:"TTBarJ_*"
DAT_002 = DIR:"./Cuts/CHN_001", FIL:"ZJJ_*"
DAT_003 = DIR:"./Cuts/CHN_001", FIL:["WWJ_*","WZJ_*","ZZJ_*"]
DAT_011 = DIR:"./Cuts/CHN_001", FIL:"MulMulJJ_*"

TRN_000 =
        INC:[
                MAS_001,MAS_003,PTM_001,PTM_002,PTM_003,ETA_001,ETA_002,ETA_003,
                MET_000,MHT_000,MEF_000,TTM_001,CTS_001,MT2_001,MT2_002,
                MDP_001,MDP_002,MDP_003,ODP_001,ODP_002,ODP_004,
                VAR_001,VAR_002,VAR_004,VAR_011,VAR_012,VAR_013,VAR_021
        ],
        TEX: [
                VAR_001, "$\tanh \vert \Delta \eta_{\ell_1 \ell_2} \vert$",
                VAR_002, "$\tanh \vert \Delta \eta_{\ell_1 j_1} \vert$",
                VAR_004, "$\tanh \vert \Delta \eta_{\ell_2 j_1} \vert$",
                VAR_011, "${P}_{\rm T}^{\ell_1}\div {/\!\!\!\!E}_{\rm T}$",
                VAR_012, "${P}_{\rm T}^{\ell_2}\div {/\!\!\!\!E}_{\rm T}$",
                VAR_013, "${P}_{\rm T}^{j_1}\div {/\!\!\!\!E}_{\rm T}$",
                VAR_021, "$(M_{\rm T2}^{100}-100) \div M_{\rm T2}^0$"
        ]

ESC_001 = KEY:MAS_001, CUT:[101,81]
ESC_002 = KEY:MET_000, CUT:75

CHN_001 = DAT:[001,002,003], LBL:0, ESC:[+001,+002]
CHN_011 = DAT:011, LBL:1, ESC:[+001,+002]

TRN_001 = CHN:[001,011]
```

Card E: Machine learning cardfile example.