Ntuple Wizard: an application to access large-scale open data from LHCb

Christine A. Aidala[1], Chris Burr[2], Marco Cattaneo[2], Dillon S. Fitzgerald[1*], Adam Morris[2,3], Sebastian Neubert[3] and Donijor Tropmann[2,4]

[1*]Department of Physics, University of Michigan, 450 Church St, Ann Arbor, 48109, Michigan, USA.
[2]European Organization for Nuclear Research (CERN), Geneva, Switzerland.
[3]Universität Bonn – Helmholtz-Institut für Strahlen und Kernphysik, Bonn, Germany.
[4]RWTH Aachen University, Aachen, Germany.

*Corresponding author(s). E-mail(s): dillfitz@umich.edu;

Abstract

Making the large data sets collected at the Large Hadron Collider (LHC) accessible to the world is a considerable challenge because of both the complexity and the volume of data. This paper presents the Ntuple Wizard, an application that leverages the existing computing infrastructure available to the LHCb collaboration in order to enable third-party users to request specific data. An intuitive web interface allows the discovery of accessible data sets and guides the user through the process of specifying a configuration-based request. The application allows for fine-grained control of the level of access granted to the public.

Keywords: Open Data, Open Access, LHCb, LHC, CERN, HEP, Outreach

1 Introduction

In an increasingly diverse research landscape, management and curation of public data are becoming critical components of transdisciplinary science. Keys to the realization of an open research ecosystem that adds scientific value have been identified in the FAIR principles of scientific data management and stewardship [1]. Making data Findable, Accessible, Interoperable, and Reusable, however, requires a considerable amount of tooling and infrastructure.

A common problem, which is acute for data in high-energy physics but increasingly an issue in other fields as well, is the sheer size of data sets stored in custom file formats. For large-scale experimental facilities, such as the LHC at the European Organization for Nuclear Research (CERN), the data sets are so large that even access by the directly involved scientists has to be centrally managed. As an example, the LHCb data collected in the years 2011-12, corresponding to $\sim 3\ \text{fb}^{-1}$ of proton-proton collisions amount to a volume of 900 TB. This volume only refers to the already preprocessed data available to members of the collaboration and scheduled for release to the public. For the purpose of processing these data, extensive computing infrastructure has been set up by the countries participating in this type of research [2]. Replicating such an infrastructure to allow the public to handle the data would not only require dedicated expert knowledge, but it would also duplicate existing facilities. On the other hand, any individual research conducted on a typical LHC data set will often only make use of a tiny portion of the full data, filtered and selected according to the requirements of the respective research question. It is therefore natural to provide the public with FAIR access to those highly selective subsamples.

In the following, an application is presented that exposes a data query service to allow the public to request sub-samples of data collected and published by the LHCb experiment. The samples are delivered as ROOT Ntuples [3] a data format that requires no special LHCb-specific software to read and for which converters to other standard file formats exist. We call the application the Ntuple Wizard.

The application interface guides users with basic knowledge in particle physics through the process of discovering the available data and formulating a useful query. The queries can be processed by the existing data production infrastructure, and results will be delivered through the CERN Open Data Portal [4]. By splitting the data request into the construction of a data query and subsequent processing of the query on the internal infrastructure, the LHCb collaboration retains fine-grained control over access to the data. Crucially this system protects the compute infrastructure from attacks by malicious code injection.

1.1 Accessible open data

In 2020, the LHC experiments at CERN adopted a new Open Data Policy [5], the scope of which expanded in 2022 to an Open Science Policy [6]. These documents define the commitments of CERN to make the data collected at the LHC, at several levels of complexity, publicly available [7]:

Level 1 Published results — this can include tables and figures but also preprocessed Ntuples or binned and unbinned fit likelihood functions.
Level 2 Outreach and education — usually in the form of highly preprocessed Ntuples.
Level 3 Reconstructed data — these data have been preprocessed to derive physics objects, such as charged particle candidates, photons, or particle jets. Reconstructed data may or may not be corrected for detector effects, such as efficiency and resolution.
Level 4 Raw data – the basic quantities recorded by the experimental instruments.

Both Level 1 and 2 data are considered to be highly processed, abstracted, and manageable using commonly available computers. Level 4 raw data will not be made available due to practical reasons concerning data size but also detector-specific information needed for the interpretation of these data. This leaves Level 3 data as the most versatile and basic data set which will be publicly accessible.

All LHC experiments have long and intricate data reconstruction pipelines, which yield several intermediate output data formats. During a pipeline, the raw data are converted to physical objects such as charged particle trajectories, jets, and vertices. Furthermore, the raw data are classified and filtered to obtain samples enriched in interesting signatures.

Figure 1 shows an overview of the data processing pipeline in LHCb as it has been used during LHC data-taking Runs 1 and 2 (2011–18). The various steps of the pipeline are outlined further in the LHCb computing technical design reports [8, 9]. Level 3 data have been defined as the output of the *stripping* step. The stripping consists of a large number of selection algorithms called *lines*, which are designed to filter the data and sort events into several collections, which are called *streams*. Streams are defined according to common physics signatures and aim to collect selections with significant overlaps into a common set of files, to reduce duplication of the data. The LHCb data organization is discussed in more detail in Appendix A, including a list of streams available in Runs 1 and 2.

The stripping selections are based on the concept of physics *candidates*. A candidate refers to a set of data matching a particular physics signature. In most cases, this signature will be a particular particle decay, such as for example $B^+ \to \bar{D}^0\pi^+$ with the subsequent decay $\bar{D}^0 \to K^+\pi^-$, where B, D, K, and π mesons are the lightest hadrons containing b, c, s, and u/d quarks respectively. Such cascading decays are represented as tree-like data structures, where the nodes represent (intermediate) particles and the edges indicate a parent-child relationship in the

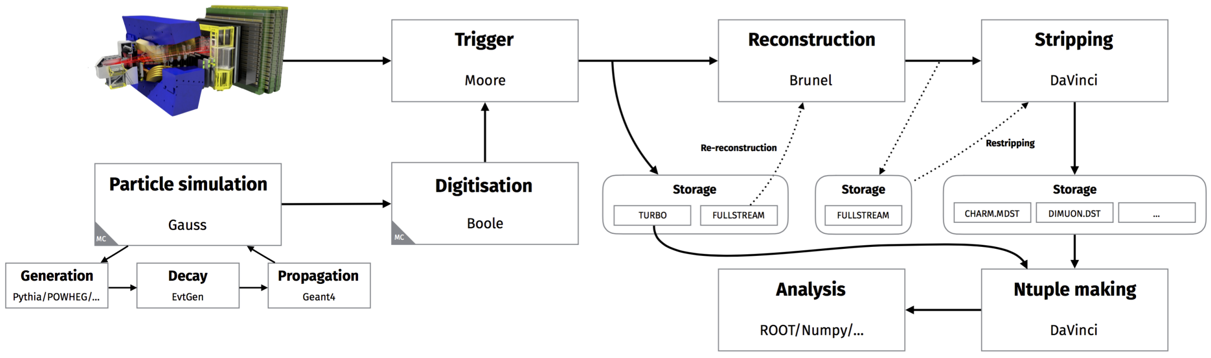


Fig. 1 LHCb data flow in Runs 1 and 2. The output of the stripping step will be made public through the CERN Open Data Portal [4].

decay. These data structures are referred to as *decay trees*. The root particle of the decay tree (the B^+ in our example) is called its *head*. Stripping selections attempt to find sets of physics objects in the reconstructed LHCb data, which match the desired decay tree and any additional criteria that might be applied to distinguish the intended signal process from background. Typical selection criteria include kinematic variables, vertex and track reconstruction qualities, and particle identification variables. Some stripping lines rely on multivariate classifiers to combine several observables into a single powerful selection criterion. The output of this procedure is collections of decay candidates specified by their particular decay trees in a custom LHCb-specific data format.

It is important to note that candidates are distinct from the concept of *events* in the LHCb data processing. An event is defined during the data acquisition and refers to a particular time window in which collisions can occur. Several proton-proton collisions can happen during this time window, and in principle, it can happen that several candidates for a particular decay are identified for a single collision. In such cases, relevant quantities (related to vertex reconstruction and flight distances) can be computed for every primary vertex (e.g. collision point) in the event.

In order to convert these data into a framework-independent format a useful concept is the aforementioned Ntuples. The idea of a Ntuple is simple: each candidate is described by a tuple of variables, i.e. physical observables of interest measured on the particular candidate, or referring to the global event in which the candidate was found. A data set consists of N such tuples, much like a simple CSV file. Ntuples are saved in ROOT files [3] and only basic data types are allowed for the variables. As a small complication in some instances, the variables can actually be arrays of basic data types. In such cases, the Ntuple Wizard provides the necessary documentation for the interpretation.

1.2 Principle of Ntuple creation and the Ntuple Wizard

Both the stripping as well as the Ntuple-making step in Fig. 1 are handled by DaVinci [8–10], an LHCb application for event selection and data analysis using the Gaudi framework [8, 9, 11]. DaVinci is configured via Python scripts and used to process entire data sets with batch processing. Both the Python configuration as well as the batch production system are intentionally hidden from users of the Ntuple Wizard for security reasons.

The DaVinci application provides access to a number of algorithms that can be combined in sequence for event selection and processing. In order to produce a Ntuple the user has to specify which variables should appear in the output data. This Ntuple configuration is handled by an algorithm named *DecayTreeTuple*, in which variables are registered through the use of so-called *TupleTools* and *LoKi functors*. A large collection of those tools and functors are available for the user to choose from. In general, a TupleTool will add a set of variables to the Ntuple, while a LoKi functor usually computes a single number. The *LoKi::Hybrid::TupleTool* can be used to write the output of functors into the tuple. Functors can be combined with standard arithmetic and logic operations, providing a flexible and powerful system to compute derived quantities. A list of important available tools is presented in Appendix B.

Figure 2 shows an overview of the Ntuple Wizard architecture, the core functionality of which is the configuration of DaVinci. The metadata and documentation describing the available data, preselections, as well as available selection operations are generated from the original provenance traces of the data and the stripping selection code. The web interface presents this metadata and documentation to the user in a pedagogical way that facilitates data discovery and formulation of the query. The query to the data has two principal parts: Data set discovery and Ntuple configuration. First, the application allows the user to select from the available predefined stripping selections, data-taking periods, and conditions. In the second step, the user defines what quantities should be computed and written to the output Ntuple. Standard tools for the computation of typical quantities, such as kinematic variables, particle identification (PID) variables, etc., are available. The query formulated by the user is stored in a set of configuration files. These files can be converted into a Python configuration compatible with the internal LHCb Analysis Productions system [12]. This conversion and the final submission of the query to the compute infrastructure are handled through an LHCb Analysis Productions manager.

1.3 Security considerations

Accepting arbitrary external code to run on the LHCb computing resources has obvious unacceptable security risks. Therefore, the Ntuple Wizard is designed to generate the configuration in a pure data-structure format. As shown in Figure 2, the configuration of the query is captured in YAML files, which can be downloaded and submitted to an LHCb Analysis Productions manager for further processing.

2 Metadata and documentation acquisition

In order to facilitate the core functionality of the Ntuple Wizard — namely data set discovery (Sec. 4) and algorithm configuration (Sec. 5), metadata and documentation from several sources are required. In particular, the application needs to know what types of decays can be queried and what tools are available to compute derived quantities of interest about these candidates.

Since these metadata are unchanging, and providing direct access to the various sources requires authentication and introduces more points of failure, the metadata are collated and served as static files over HTTP. No additional access to the LHCb code or database is needed by the Ntuple Wizard once it has been deployed.

The sources of metadata can be grouped into two coarse categories: the LHCb software stack and the LHCb database. Metadata are acquired from the LHCb software stack in two ways. The first is from the Gaudi Python interface; particularly under the DaVinci application environment. Metadata about the configuration interface of each TupleTool are extracted from DaVinci. Details of the stripping lines, including the chain of selection algorithms that define them, are extracted from the DaVinci versions used in the corresponding stripping campaigns.

The process of building decay candidates in a stripping line often involves a combination of many algorithms from the LHCb selection framework, which combine particles, impose selection requirements, perform PID substitution, and build final-state particle candidates from trajectories of charged particles and calorimeter clusters. The algorithms can be related to each other through their input and output locations. The full list of decays (including all sub-decays) must be inferred by traversing the 'dependency tree' of the selection algorithms. This is performed using custom code during metadata acquisition.

The second, more indirect way is from the LHCb Doxygen pages, which themselves are generated from the source code of the LHCb software stack. The latest Doxygen pages for Run 1 or Run 2 DaVinci versions are used to extract the documentation for each TupleTool and LoKi functor. A campaign to improve the Doxygen documentation at its source was undertaken during the development of the Ntuple Wizard.

The LHCb database provides metadata about the centrally managed data sets, which is necessary to configure the Ntupling productions as explained above. In order not to duplicate effort,

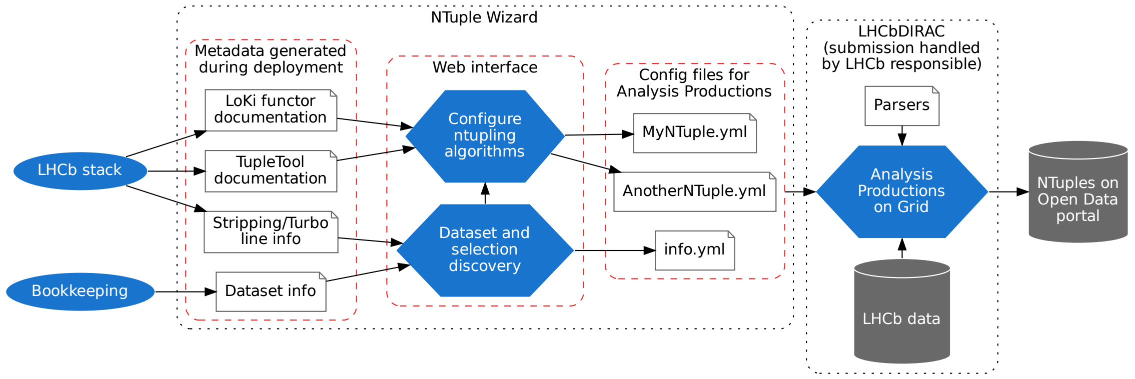

NTuple Wizard
Metadata generated during deployment
LoKi functor documentation
TupleTool documentation
Stripping/Turbo line info
Dataset info
LHCb stack
Bookkeeping
Web interface
Configure ntupling algorithms
Dataset and selection discovery
Config files for Analysis Productions
MyNTuple.yml
AnotherNTuple.yml
info.yml
LHCbDIRAC (submission handled by LHCb responsible)
Parsers
Analysis Productions on Grid
LHCb data
NTuples on Open Data portal

Fig. 2 Architecture of the Ntuple Wizard.

a common code base is employed to extract metadata from the LHCb database for both the Ntuple Wizard and the CERN Open Data Portal.

3 User interface

The user interface consists of a sequence of dialogues that guide the user through the configuration steps. This is designed as a client-side dynamic web page that reads metadata acquired at deployment time to serve as static files (see Sec. 2).

Since users of LHCb open data do not, in general, have access to the same support network of experienced users and developers enjoyed by LHCb collaboration members, a key design element of the Wizard is to provide the necessary documentation for a novice user to complete each step of the configuration process.

The existing documentation of DaVinci [8–10] is fragmented across several sources (Twiki [13], the Starterkit [14], Doxygen [15] and the source code itself), so where possible, the Wizard pulls text from each of these disparate sources and renders it in the relevant context within the user interface.

There are two main steps to formulate a query using the Ntuple Wizard: Dataset discovery and Ntuple configuration. These steps are explained in the following.

4 Dataset discovery and production configuration

The available data contain a wide range of stripping selections, data-taking periods, and running conditions. The **Production configuration** dialogue of the Ntuple Wizard guides the user through the selection of the desired subsets. The interface allows the selection of several decays to be processed simultaneously as part of one query. For each decay, a separate Ntuple will be produced.

4.1 Discovering available candidate decays

In the **Decay search** dialogue, the Ntuple Wizard presents a list of all decays selected by the stripping, accompanied by decay descriptors in LoKi and LaTeX formats, information about which stripping lines build them, as well as 'tags' that can be used to filter different types of decays. Decays are searchable through various filters, including the identity or properties of the parent particle and decay products, whether the candidates are built by a specific stripping line, and the aforementioned tags. An example of the decay search is shown in Figure 3. The selected candidate of interest is highlighted in blue, and the collection was narrowed down from the list of all possible decays by using the filters and tags at the top of the page. The 'none of' option of the tags drop-down menu is chosen by default, indicating that decays with the displayed tags are hidden from the list of selectable decays. The tags 'charge-violating' and 'undefined-unstable' corresponding to decays that violate charge conservation and that contain unstable particles without defined decays respectively are hidden by default. If the user wishes to instead isolate decays that meet the criteria of a given tag, a different option can be selected from the 'tags' drop-down menu. It is possible to select several decays for further processing at this stage.

4.2 Stripping line and data set selection

Once a decay is selected by the user, all corresponding stripping lines and data sets from the various running periods are listed, and the desired combination(s) can be selected. The case can arise where the same stripping line shows up in multiple stripping versions within the same dataset (stream, running year, and magnet polarity). These are rendered as separate options in the dataset selection drop-down menu of the Ntuple Wizard. For a given decay, it is recommended to choose only one dataset for each magnet polarity within a given running year, and to use the most recent stripping version in the case of duplicates. The data organization of LHCb is elaborated on in Appendix A, including a table of running years, as well as corresponding collision energies and stripping versions.

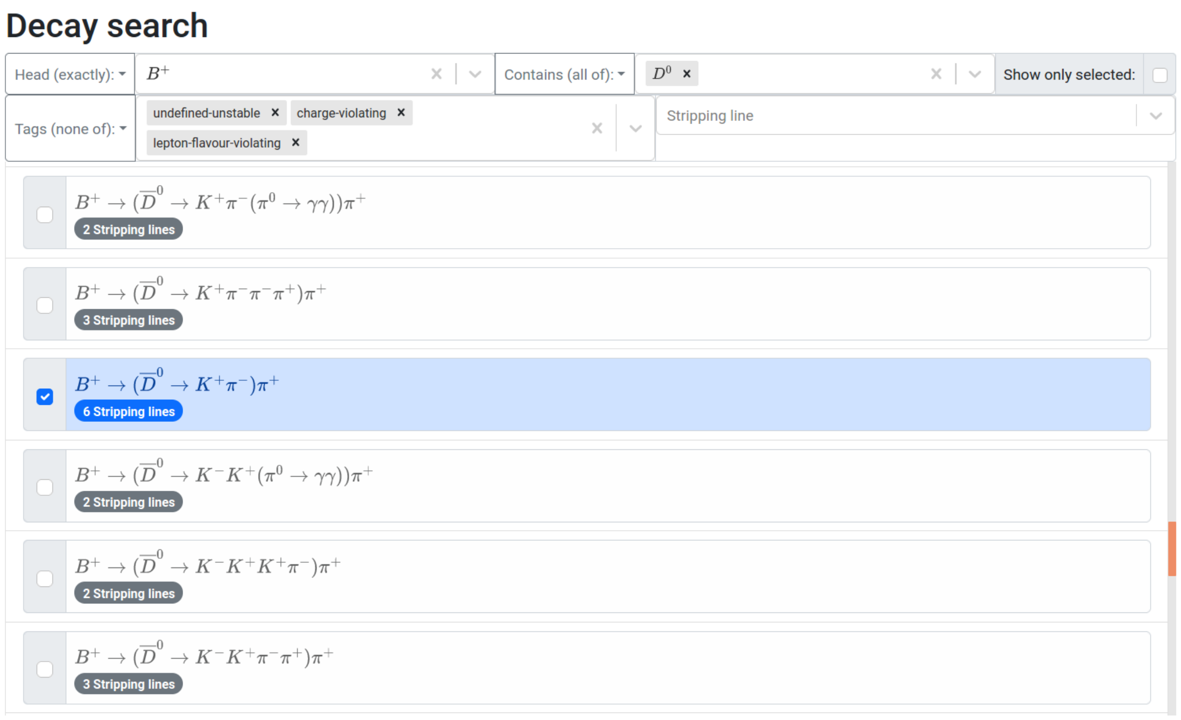
Links to documentation about each stripping line including selection algorithms that went into building the decay candidates are displayed to the user to guide them in choosing the most suitable stripping line and data stream for their physics interest. Figure 4 shows an example of the production configuration page, where an available stripping line and data set have been chosen from lists of all possibilities corresponding to the selected decay channel. The blue question mark button contains links to the aforementioned stripping documentation. At this point, the query is specified up to deciding what information to write into the Ntuple.

5 Ntuple configuration

The **DecayTreeTuple configuration** dialogue is designed to guide the user through customization of the quantities written to the Ntuple for the selected candidates. For each decay, a separate DecayTreeTuple has to be configured. Care should be taken to name the Ntuples appropriately. The Ntuple Wizard requires a unique name for each Ntuple.

Selected decay trees are visually represented as graphs, where each physics object (e.g. particle) is represented by a node as shown in the screenshots in Figure 5. The user can interact with this graph by selecting one or multiple nodes at a time and determining which TupleTools will be added to each node, which in turn determines which quantities are saved to the Ntuple. A list is rendered on screen depending on the selected node(s), each element of which corresponds to a selected TupleTool, with buttons for configuring and removing the tool. The TupleTool configuration interface includes links to relevant documentation about the tool, including lists of quantities written by the tool where available. Each node in the graph comes with the standard set of TupleTools for LHCb analyses, but more will often be needed depending on the particular physics interests of the user. Furthermore, any added tool will come with the standard configuration, which can be further modified if the user desires. A custom set of standard LoKi variables and functions of these variables can also be saved to the Ntuple for each node, using the *Loki::Hybrid::TupleTool*. Appendix B contains a brief description of the standard set of TupleTools included with each node on the graph, as well as other useful TupleTools for physics analysis. Figure 5 shows an example of the configurable graph corresponding to the selected candidate shown in Figures 3 and 4, as well as a list of TupleTools corresponding to the entire decay candidate (top), and particular nodes selected on the graph (bottom). It can be seen from the figure that nodes can also be selected through the categories shown below the graph and that TupleTools can be added, removed, or configured for each node or grouping of nodes.

Figure 6 shows an example of the user interface for configuring TupleTools, with the particular example showing *TupleToolTISTOS*, which saves trigger information to the Ntuple. It can be seen at the bottom how relevant information is provided.


Decay search

Head (exactly): B^+

Contains (all of): D^0

Show only selected:

Tags (none of): undefined-unstable, charge-violating, lepton-flavour-violating

Stripping line

$B^+ \to (\overline{D}^0 \to K^+\pi^-(\pi^0 \to \gamma\gamma))\pi^+$ 2 Stripping lines

$B^+ \to (\overline{D}^0 \to K^+\pi^-\pi^-\pi^+)\pi^+$ 3 Stripping lines

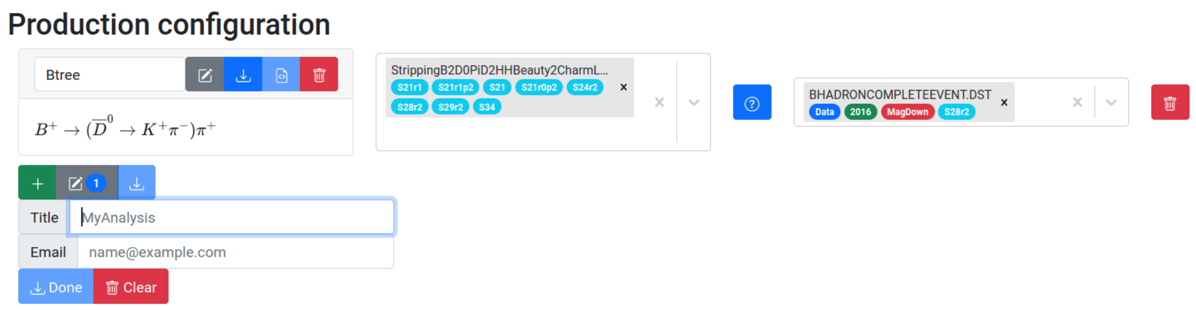
$B^+ \to (\overline{D}^0 \to K^+\pi^-)\pi^+$ 6 Stripping lines

$B^+ \to (\overline{D}^0 \to K^-K^+(\pi^0 \to \gamma\gamma))\pi^+$ 2 Stripping lines

$B^+ \to (\overline{D}^0 \to K^-K^+K^+\pi^-)\pi^+$ 2 Stripping lines

$B^+ \to (\overline{D}^0 \to K^-K^+\pi^-\pi^+)\pi^+$ 3 Stripping lines

Fig. 3 Example of the decay candidate search function of the Ntuple Wizard.


Production configuration

Btree

$B^+ \to (\overline{D}^0 \to K^+\pi^-)\pi^+$

StrippingB2D0PiD2HHBeauty2CharmL... S21r1 S21r1p2 S21 S21r0p2 S24r2 S28r2 S29r2 S34

BHADRONCOMPLETEEVENT.DST Data 2016 MagDown S28r2

Title: MyAnalysis

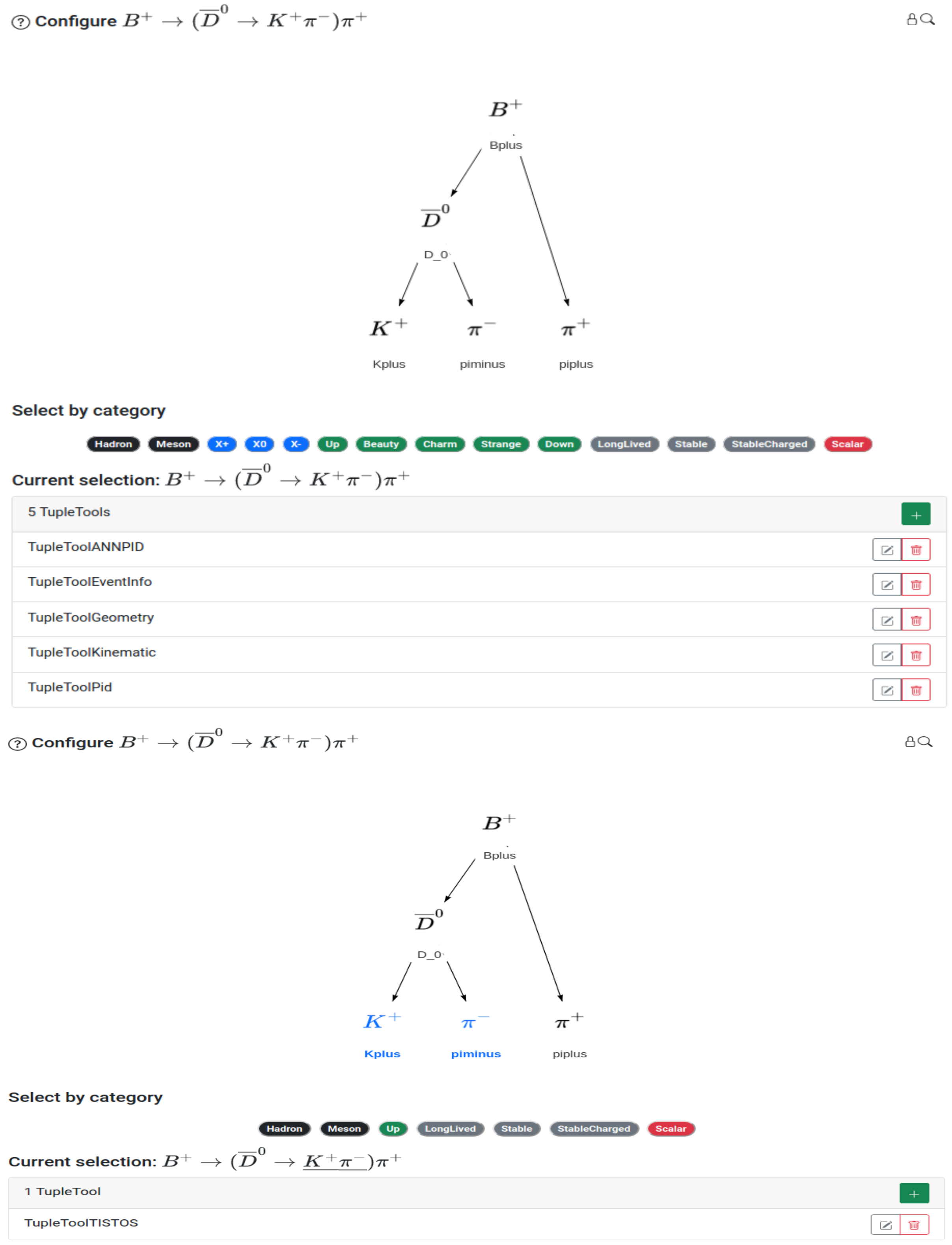
Email: name@example.com

Done Clear

Fig. 4 Example of the data set selection and production configuration step of the Ntuple Wizard.

5.1 Configuration output

Figure 7 shows an example of the output YAML file used to configure the DecayTreeTuple algorithm that was populated via configurations captured in Figs. 5 – 6, where the `tools`, `groups` and `branches` keys are shown specifying which TupleTools and therefore which information will be saved to the Ntuple. The top-level key `tools` contains a list of TupleTool configurations, from which the parsing functions create and configure TupleTool algorithms attached to the DecayTreeTuple itself, which will thus write either particle-level information about the decay or event-level information, depending on the class of the TupleTool. The keys `branches` and `groups` themselves contain lists of dictionaries whose keys specify particles and have their own `tools` lists which are used similarly to attach TupleTool algorithms to the specified particle(s) in the decay tree. Note that `groups` differs from `branches` in that it specifies multiple particles to be looped over and have identically configured TupleTool algorithms attached.



Current selection: $B^+ \to (\overline{D}^0 \to K^+\pi^-)\pi^+$

5 TupleTools

TupleToolANNPID

TupleToolEventInfo

TupleToolGeometry

TupleToolKinematic

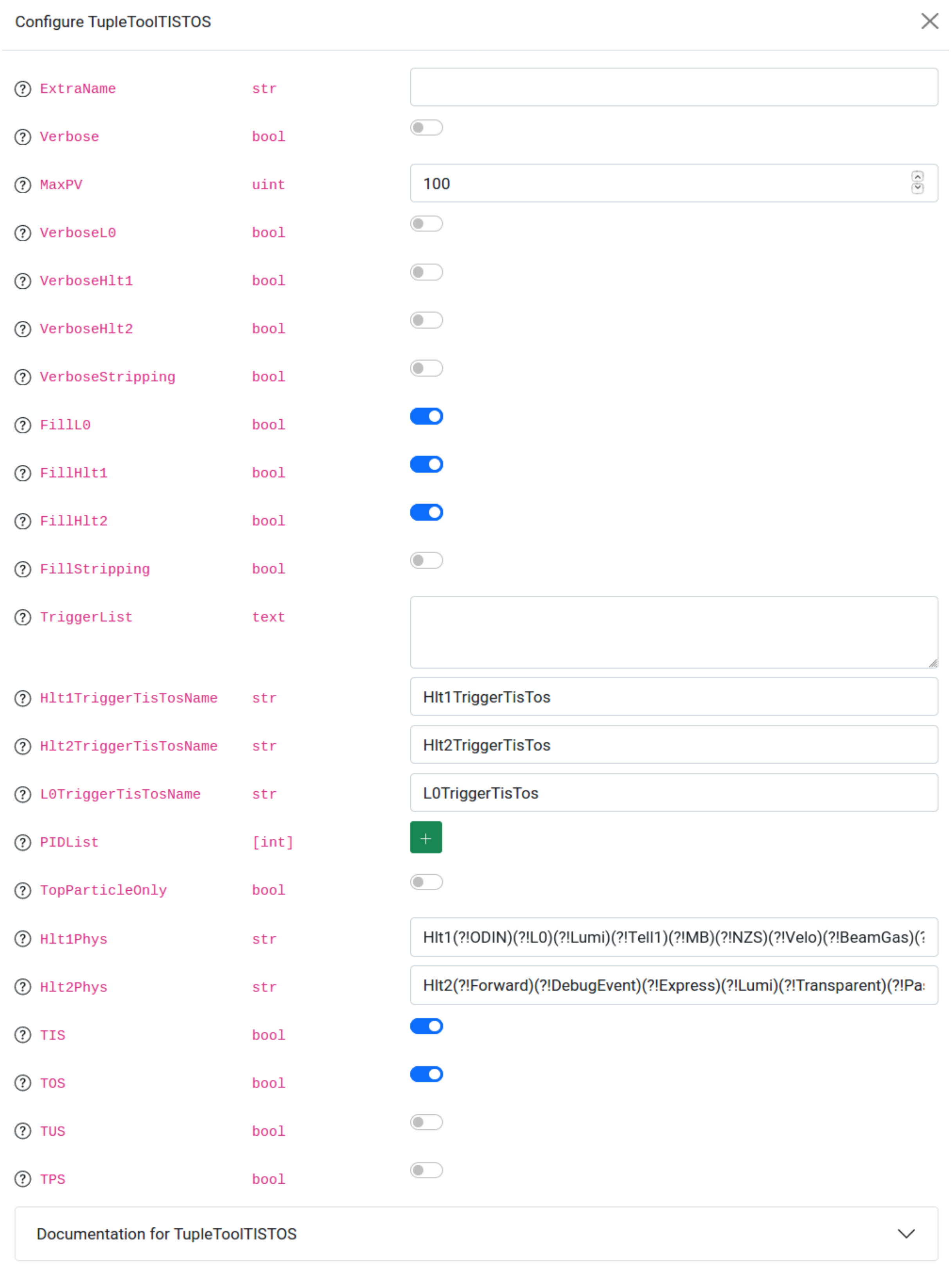
TupleToolPid

Current selection: $B^+ \to (\overline{D}^0 \to \underline{K^+\pi^-})\pi^+$

1 TupleTool

TupleToolTISTOS

Fig. 5 Example of an interactive graph used to configure DecayTreeTuple, with selected TupleTools displayed for both the entire candidate (top) and selected nodes (bottom).



Configure TupleToolTISTOS

ExtraName	str	
Verbose	bool	○
MaxPV	uint	100
VerboseL0	bool	○
VerboseHlt1	bool	○
VerboseHlt2	bool	○
VerboseStripping	bool	○
FillL0	bool	●
FillHlt1	bool	●
FillHlt2	bool	●
FillStripping	bool	○
TriggerList	text	
Hlt1TriggerTisTosName	str	Hlt1TriggerTisTos
Hlt2TriggerTisTosName	str	Hlt2TriggerTisTos
L0TriggerTisTosName	str	L0TriggerTisTos
PIDList	[int]	+
TopParticleOnly	bool	○
Hlt1Phys	str	Hlt1(?!ODIN)(?!L0)(?!Lumi)(?!Tell1)(?!MB)(?!NZS)(?!Velo)(?!BeamGas)(
Hlt2Phys	str	Hlt2(?!Forward)(?!DebugEvent)(?!Express)(?!Lumi)(?!Transparent)(?!Pa
TIS	bool	●
TOS	bool	●
TUS	bool	○
TPS	bool	○

Documentation for TupleToolTISTOS

Fig. 6 Example of the configuration interface of a TupleTool within the Ntuple Wizard, (in particular, *TupleToolTISTOS* for saving trigger information), including links to relevant documentation at the bottom of the modal.

Fig. 7 Output of Btree.yaml, the data file used to configure the DecayTreeTuple algorithm.

```
inputs:
  - /Event/BhadronCompleteEvent/Phys/B2D0PiD2HHBeauty2CharmLine/
     Particles
descriptorTemplate: ${Bplus}[B+ -> ${D_0}(D~0 -> ${Kplus}K+ ${piminus}pi
   -)${piplus}pi+]CC
tools:
  - TupleToolKinematic:
      ExtraName: ''
      Verbose: false
      MaxPV: 100
      Transporter: ParticleTransporter:PUBLIC
  - TupleToolPid:
      ExtraName: ''
      Verbose: false
      MaxPV: 100
  - TupleToolANNPID:
      ExtraName: ''
      Verbose: false
      MaxPV: 100
      ANNPIDTunes:
        - MC12TuneV2
        - MC12TuneV3
        - MC12TuneV4
        - MC15TuneV1
      PIDTypes:
        - Electron
        - Muon
        - Pion
        - Kaon
        - Proton
        - Ghost
  - TupleToolGeometry:
      ExtraName: ''
      Verbose: false
      MaxPV: 100
      RefitPVs: false
      PVReFitter: LoKi::PVReFitter:PUBLIC
      FillMultiPV: false
  - TupleToolEventInfo:
      ExtraName: ''
      Verbose: false
      MaxPV: 100
branches:
  Bplus:
    particle: B+
    tools: []
  D_0:
    particle: D~0
    tools: []
  Kplus:
    particle: K+
    tools: []
  piminus:
    particle: pi-
    tools: []
  piplus:
    particle: pi+
    tools: []
groups:
  Kplus,piminus:
    particles:
      - K+
      - pi-
    tools:
      - TupleToolTISTOS:
          ExtraName: ''
          Verbose: false
          MaxPV: 100
          VerboseL0: false
          VerboseHlt1: false
          VerboseHlt2: false
          VerboseStripping: false
          FillL0: true
          FillHlt1: true
          FillHlt2: true
          FillStripping: false
          TriggerList: []
          Hlt1TriggerTisTosName: Hlt1TriggerTisTos
          Hlt2TriggerTisTosName: Hlt2TriggerTisTos
          L0TriggerTisTosName: L0TriggerTisTos
          PIDList: []
          TopParticleOnly: false
          Hlt1Phys: >-
            Hlt1(?!ODIN)(?!L0)(?!Lumi)(?!Tell1)(?!MB)(?!NZS)(?!Velo)(?!
               BeamGas)(?!Incident).*Decision
          Hlt2Phys: >-
            Hlt2(?!Forward)(?!DebugEvent)(?!Express)(?!Lumi)(?!
               Transparent)(?!PassThrough).*Decision
          TIS: true
          TOS: true
          TUS: false
          TPS: false
name: DecayTreeTuple/Btree
```

5.2 Future developments

It is planned to extend the current functionality of the Ntuple Wizard by including the ability to create custom candidates from standard collections of LHCb particles. Another important planned addition is the ability to configure custom jet reconstruction. Ideally, support will be included for the full set of algorithms available in DaVinci for data analysis and event/candidate selection as resources allow.

As of Run 3, which started in 2022, the majority of the filtering and preselection of the data will be done in real time within the LHCb high-level trigger (HLT). In this architecture, the data will be fully reconstructed online and the final preselection algorithms will run in the HLT. Offline preselections will be feasible for a subset of the events. In both cases the output will have the same level of abstraction as the output of the stripping, allowing for a relatively simple adaptation of the Ntuple Wizard once the Run 3 data are made public.

6 Request submission and execution

Once the candidate(s) of interest, data set(s), and information to be saved in the Ntuple(s) are specified, and a name and email address have been provided for the production, a ZIP format file containing all relevant output files for the data query can be downloaded (as shown in Figure 8) and submitted to an LHCb Analysis Productions manager.

Requests for Ntuple creation are handled using the Analysis Productions package. The files describing a new request are committed to a repository hosted on the CERN GitLab [16], and a merge request is created once they are ready for review. The Continuous Integration feature of GitLab is used to submit test productions to LHCbDIRAC [8, 9], which automatically processes a small fraction of the data when the remote repository is updated.

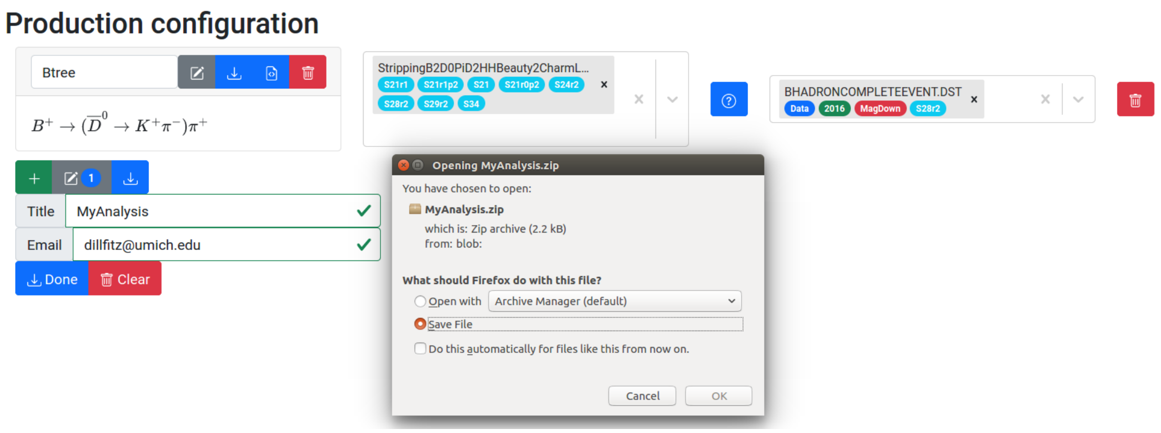
Once the request is submitted, it is handled by the LHCbDIRAC production system. A production defines how a dataset is to be processed, and LHCbDIRAC will launch and manage computing jobs until the dataset is fully processed. Productions are defined in ‘steps’ that specify which application to run and which configuration files to read, and may be chained together such that the output of the first step is the input to the second, etc. The `info.yaml` file produced by the Ntuple Wizard defines one production per dataset, each consisting of a single DaVinci step.

Within the production jobs, DaVinci is configured by functions defined in an external Python module according to the YAML files produced by the Ntuple Wizard. The data structure configured in Section 5 and displayed in Figure 7 is traversed, and the configurable properties of the DecayTreeTuple algorithm are assigned the corresponding values.

After the Analysis Production jobs are complete, the produced Ntuples will be delivered to the CERN Open Data Portal for retrieval.

7 Summary

Providing public access to the large data sets at the LHC is a significant technical challenge, but it is becoming increasingly important for the longevity of high-energy physics in order to optimize acquired knowledge from the collected data. The volume and complexity of the data collected at LHCb make providing direct access to reconstructed (Level 3) data suitable for physics research difficult, motivating the design of the Ntuple Wizard, where users can submit queries to obtain skimmed data samples (Ntuples) of the reconstructed data suitable for their physics interests. The Ntuple Wizard is a web-based application that intuitively guides the user through specifying a query, from discovering a data set from a physics candidate (e.g. decay) of interest, to configuring the information to be saved in the output Ntuple. The output of the Ntuple Wizard is a pure data structure (YAML) format, which is to be submitted to an LHCb Analysis Productions manager so it can be parsed internally to provide the necessary Python scripts needed to configure the DaVinci application. The Ntuples will ultimately be delivered to the CERN Open Data Portal for retrieval.



Production configuration

Btree

$B^+ \to (\overline{D}^0 \to K^+\pi^-)\pi^+$

Title: MyAnalysis ✓

Email: dillfitz@umich.edu ✓

StrippingB2D0PiD2HHBeauty2CharmL... S21r1 S21r1p2 S21 S21r0p2 S24r2 S28r2 S29r2 S34

BHADRONCOMPLETEEVENT.DST Data 2016 MagDown S28r2

Opening MyAnalysis.zip

You have chosen to open:

MyAnalysis.zip

which is: Zip archive (2.2 kB)

from: blob:

What should Firefox do with this file?

Open with Archive Manager (default) [✓ Save File]

Do this automatically for files like this from now on.

Fig. 8 Example of downloading the output files of the Ntuple Wizard after the query is fully specified.

Appendices

A LHCb Data Organization

Table 1 shows a list of running years, including corresponding collision energies and stripping versions available in the Ntuple Wizard.

Table 1 Table of running years, including collision energy ($\sqrt{s}$) and relevant stripping versions available in the Ntuple Wizard.

Running Year	$\sqrt{s}$ (TeV)	Stripping Versions
Run 1		
2011	7	s21r1, s21r1p1, s21r1p2
2012	8	s21, s21r0p1, s21r0p2
Run 2		
2015	13	s24r2
2016	13	s28r2, s28r2p1
2017	13	s29r2, s29r2p1, s29r2p2
2018	13	s34, s34r0p1, s34r0p2

LHCb data streams come in two formats, Data Summary Tape (DST) files, which contain the full event information, and micro Data Summary Tape (MDST) files, which only contain the physics objects directly matched in at least one stripping selection. In MDST streams, the rest of the information in the events, apart from a few global event properties, is discarded. Table 2 shows a list of streams from Run 1 and Run 2, with the DST vs MDST format indicated in the stream name.

Table 2 Table of data streams from Runs 1 and 2 available through the Ntuple Wizard.

Stream
BHADRON.MDST
BHADRONCOMPLETEEVENT.DST
CALIBRATION.DST
CHARM.MDST
CHARMCOMPLETEEVENT.DST
CHARMTOBESWUM.DST
DIMUON.DST
EW.DST
LEPTONIC.MDST
MINIBIAS.DST
PID.MDST
RADIATIVE.DST
SEMILEPTONIC.DST

B List of useful TupleTools

B.1 Default TupleTools

A set of TupleTools is included by default for all Ntuple configuration files produced by the Ntuple Wizard. Tools can be removed by the user if desired, but are standard tools used in LHCb analyses, and are recommended to keep for physics analyses. Given the flexible data structure of the Ntuple, it is easy to produce a reduced data structure with a subset of variables at a later stage in data processing, while still maintaining the full set of variables in the original Ntuple.

- *TupleToolANNPID* — A tool used to add artificial neural network particle identification information about the physics candidate to the Ntuple.
- *TupleToolEventInfo* — A tool used to add event and run information to the Ntuple.

- *TupleToolGeometry* — A tool used to add information about physics candidate geometry and event geometry to the Ntuple.
- *TupleToolKinematic* — A tool used to add kinematic information about the physics candidate to the Ntuple.
- *TupleToolPid* — A tool used to add particle identification information about the physics candidate to the Ntuple, with additional information than that in *TupleToolANNPID*, including information about which PID detector subsystems were used in the probability calculations.

B.2 Other useful TupleTools

- *TupleToolTISTOS* — A tool that saves the trigger TIS/TOS (Trigger independent of Signal/Trigger on Signal) decisions for each particle to the Ntuple.
- *LoKi::Hybrid::TupleTool* — A tool that allows the user to add LoKi variables or expressions of these variables known as LoKi functors to the Ntuple.

Acknowledgements

We thank our colleagues at LHCb for providing the necessary data and software, and within the Data Processing & Analysis (DPA) project for their incredibly valuable discussions. We additionally would like to thank Jose Marco Arias for systematic testing of the web interface. All authors acknowledge support from CERN. In addition, C.A.A. and D.S.F. acknowledge support from the National Science Foundation under Award Number 2012926, and A.M. and S.N. acknowledge support from the DFG Grant NE 2185/1-1.

References

[1] M. D. Wilkinson et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018, March 2016.

[2] I. Bird et al. Update of the Computing Models of the WLCG and the LHC Experiments. (CERN-LHCC-2014-014, LCG-TDR-002), Apr 2014.

[3] I. Antcheva et al. ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization. *Comput. Phys. Commun.*, 180:2499–2512, 2009.

[4] CERN Open Data Portal. https://opendata.cern.ch/.

[5] CERN Open Data Policy for the LHC Experiments. (CERN-OPEN-2020-013), Nov 2020. https://cds.cern.ch/record/2745133.

[6] CERN Open Science Policy. (CERN-OPEN-2022-013), 2022. https://cds.cern.ch/record/2835057.

[7] Z. Akopov et al. Status Report of the DPHEP Study Group: Towards a Global Effort for Sustainable Data Preservation in High Energy Physics. (DPHEP-2012-001, FERMILAB-PUB-12-878-PPD, arXiv:1205.4667), May 2012.

[8] The LHCb Collaboration. LHCb computing: Technical Design Report. (CERN-LHCC-2005-019; LHCb-TDR-11), 2005. http://cds.cern.ch/record/835156.

[9] The LHCb Collaboration. Upgrade Software and Computing. (CERN-LHCC-2018-007, LHCB-TDR-017), 2018. https://cds.cern.ch/record/2310827.

[10] DaVinci. https://lhcbdoc.web.cern.ch/lhcbdoc/davinci/.

[11] Gaudi Project. https://gaudi.web.cern.ch/gaudi/.

[12] M. Ferrillo. New generation offline software for the LHCb upgrade I. 2022. https://cds.cern.ch/record/2806414.

[13] LHCb Twiki. https://twiki.cern.ch/twiki/bin/view/LHCb/.

[14] LHCb Starterkit. https://lhcb.github.io/starterkit-lessons/index.html.

[15] LHCb Doxygen. https://lhcb.web.cern.ch/computing/Support/Doxygen/doxygen.htm.

[16] GitLab. https://gitlab.cern.ch/.