Towards a Standardised Data Release Format

Luke Pickering
Rutherford Appleton Laboratory, STFC

P. Stowell (Sheffield)
NuXTract, CERN, 2023/10/04

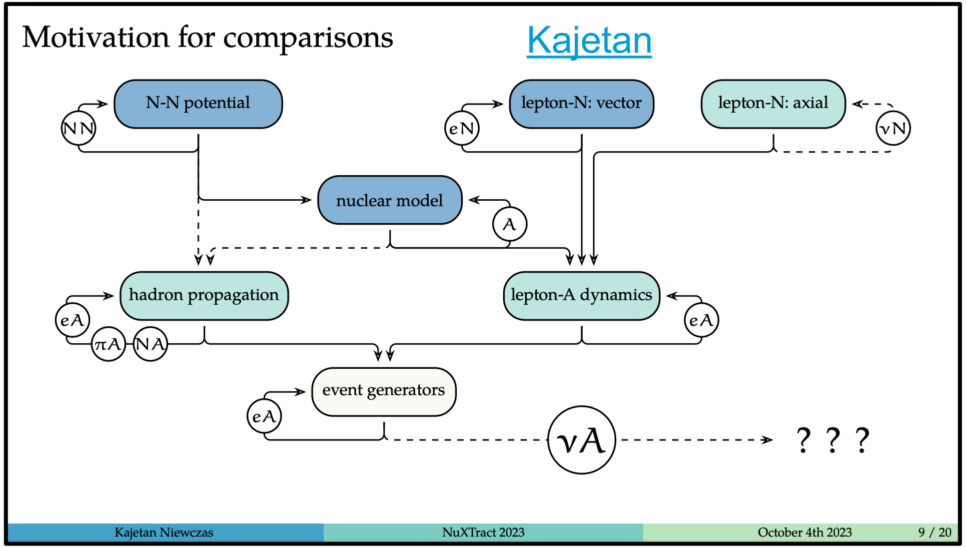
Science and Technology Facilities Council

Supported by URF\R1\211661
The Royal Society

whoami

- T2K Neutrino-Interactions Worrier
- DUNE(-PRISM) PMNS Oscillations
- NUISANCE
  - My biases/experience mainly come from here as a ‘consumer’ of data releases
  - I am starting this effort to provide a new set of core tools and standards for NUISANCE but if we can get community buy-in we can provide tools for ‘automating’ data—theory comparisons.

The Problem

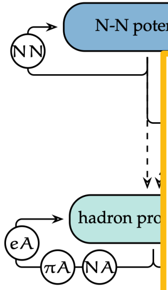


Motivation for comparisons
Kajetan
N-N potential
lepton-N: vector
lepton-N: axial
NN
eN
νN
nuclear model
A
hadron propagation
lepton-A dynamics
eA
πA
NA
eA
event generators
eA
νA
? ? ?
Kajetan Niewczas
NuXTract 2023
October 4th 2023
9 / 20

The Problem

Motivation for comparisons

Kajetan



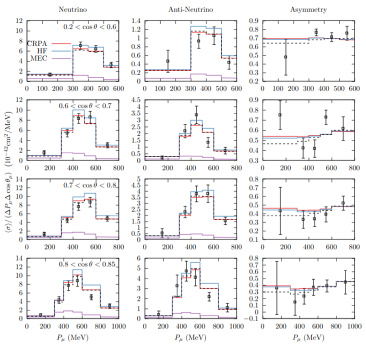
Kajetan Niewczas

Uses for neutrino data

Oscillation/BSM analysis → not discussed

Fitting & parameter searches

What I use data for:



[Arxiv:2110.11321] Alexis

[S. Dolan e.a. 2110.14601]

	Carbon and oxygen	ν_μ and $\bar{\nu}_\mu$
Number of bins	58	116
HF-CRPA	135	740
HF	143	683
SuSAv2	140	741
LFG-RPA	59	446
LFG (no RPA)	184	1028

Data-model comparisons

It's nice to have a number…

But doesn't teach me much

Fermilab

4 Experimental data from a theoretical needs POV | NuXTract 2023, 4 October 2023, CERN

The Problem

Motivation for comparisons Kajetan

N-N potential

NN

lepton-N: vector

lepton-N: axial

hadron pro

eA

πA

NA

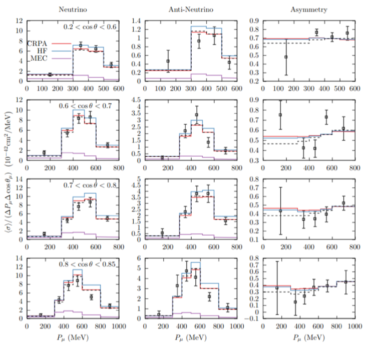
Kajetan Niewczas

Uses for neutrino data

Oscillation/BSM analysis → not discussed

Fitting & parameter searches

What I use data for:



Number of bi
HF-CRPA
HF
SuSAv2
LFG-RPA
LFG (no RPA

Data-model

It's nice to h

But doesn't

[Arxiv:2110.11321] Alexis

Data points

- **Data releases** are the main source of acquiring modern measurements

e.g. Data is usually released in the **ROOT format**
→ **theoreticians do not use ROOT**
→ this forces us to write special scripts to extract a few datapoints
→ extracting more sophisticated information is troublesome

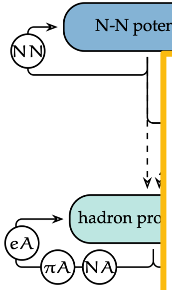
Q: **Is it possible to always provide data in a txt-like file?**

How will we manage this once we approach multi-dimensional data?

Is is possible to provide scripts providing minimal working examples?

The Prob

Motivation for compar



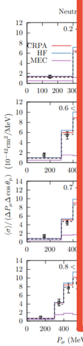
Kajetan Niewczas

Uses fo

Oscilla

Fitting

What I



[Arxiv:2110.11321]

Fermilab

General ideas, questions and suggestions

- Provide CS data, covariance and smearing matrices in ASCII files (.txt, .dat, .csv).
- Include a reference into the article for flux and data release, specifying cuts, thresholds, etc.
- What should be included in χ2 analyses? Covariance and smearing matrices: state clearly how and when to apply them.
- How to calculate some cross sections or how can be obtained, e.g. unfolded MicroBooNE σ/<Ev> vs. Ev
- Could be applied the additional smearing matrix to the data? That could ease the theory-vs-data comparison (for us) or a wrong use of Ac by theorists. Conexion with forward-folding?
- It would be also interesting to have more CC1π data in terms of lepton or proton kinematics apart from pion kinematics.
- E_had? E_cal? E_avail? Being able to fully analyze exp. data without need of implementing models in generators.

Guillermo

Some Terminology

- Analyser: Experimental collaborator, generally who makes selections, assesses errors, produces XS data release, contributes to paper writing
  - Expert in the measurement
- Consumer: Theoretician or model tuner, Compare model or generator predictions to one or more measurement
  - Not an expert in a given measurement
- User: A third-party making use of a consumer framework, but not a core developer themselves
  - Not an expert in the measurement or the comparison software
- Automatic Comparison: A data—MC comparison that can be performed without measurement-specific custom code on the Consumer or User’s behalf.

Goal of This Talk

- Start the process of developing a standard for XS data releases:
  - Currently a lot of what is required to predict a measurement lives in the meta-analysis tools themselves (NUISANCE, GENIE Comparisons, Theoretician analyses)
  - High-level aim is to separate these bits out so that they are accessible to all and provided by the *analysers*
  - **Win-Win:** Makes both the *analysers* and *consumer's* lives easier and responsibilities clearer
  - No loss of functionality or extensibility

Goal of This Talk

- Start the process of developing a standard for XS data releases:
  - Currently a lot of what is required to predict a measurement lives in the meta-analysis tools themselves (NUISANCE, GENIE Comparisons, Theoretician analyses)
  - High-level aim is to separate these bits out so that they are accessible to all and provided by the *analysers*
  - **Win-Win:** Makes both the *analysers* and *consumer's* lives easier and responsibilities clearer
  - No loss of functionality or extensibility

- This needs to be a process, upcoming draft standards that I talk about here will need engagement and feedback from *analysers* and *consumers:*
  - This is a difficult problem, with many considerations, but it is possible to standardize it enough to improve the situation without constraining what kinds of data can be presented.
  - I'm not trying to solve the problem in a vacuum!

Automatic Comparisons

What Is Needed to Predict a Cross Section Measurement?

- A Signal definition
- Event projection operators

Signal Definition And Event Projection Operators

- A Signal definition
  - An unambiguous description of what is considered a 'signal' interaction
  - This is ***not*** the same as the event ***selection*** that an analyser makes when looking at reconstructed information
  - Reconstructed interactions that are ***selected*** but are not considered ***signal*** must be treated as background

Signal Definition And Event Projection Operators

- A Signal definition
  - An unambiguous description of what is considered a 'signal' interaction
  - This is ***not*** the same as the event ***selection*** that an analyser makes when looking at reconstructed information
  - Reconstructed interactions that are ***selected*** but are not considered ***signal*** must be treated as background
  - For the vast majority of the measurements NUISANCE handles, background treatment has already been applied:
    - Events that fall within the Signal Definition contribute to a measurement
    - Events that fall without do not contribute to a measurement

Signal Definition And Event Projection Operators

- A Signal definition
  - An unambiguous description of what is considered a 'signal' interaction
  - This is ***not*** the same as the event ***selection*** that an analyser makes when looking at reconstructed information
  - Reconstructed interactions that are ***selected*** but are not considered ***signal*** must be treated as background
  - For the vast majority of the measurements NUISANCE handles, background treatment has already been applied:
    - Events that fall within the Signal Definition contribute to a measurement
    - Events that fall without do not contribute to a measurement

- Event projection operators:
  - Might be called 'independent variables' or kinematic observables
  - The event properties that a cross section is measured as a function of: e.g. Pmu, Q2QE, SumTProt
  - Often these are unsmeared to true quantities as part of the measurement

Signal Definition And E

- A Signal definition
  - An unambiguous description of
  - This is ***not*** the same as the ev … cted information
  - Reconstructed interactions that … background
  - For the vast majority of the mea … dy been applied:
    - Events that fall within the Signal Definition contribute to a measurement
    - Events that fall without do not contribute to a measurement

PRL.129.021803

```
README

Data release README. All files are contained in the associated compressed
tarball - MINERvA_TripleDiffQELike_DataRelease.tar.gz

Signal definition-

Kinematic windows:

P|| > 1.5 GeV
Muon Angle w.r.t neutrino < 20 degrees


Interaction and final state information:

Charge Current interaction of a muon type neutrino with the following
particles allowed in the final state:
muon, any number of nucleons, gammas <10 MeV (nuclear deexcitation)
```

PRL 129 021803

- Event projection operators:
  - Might be called 'independent variab
  - The event properties that a cross se … SumTProt
  - Often these are unsmeared to true

```
###
SumTp vs q0_qe vs Muon E
###
----
```

What Is Needed to Predict a Cross Section Measurement?

- A Signal definition
- Event projection operators

What Is Needed to Predict a Cross Section Measurement?

- A Signal definition
- Event projection operators
- Measured data points
- Uncertainty descriptions
  - Errors on each bin
  - Covariance matrices across bins and measurements
  - Errors broken down by source
- Measurement metadata:
  - Some is important for making predictions: Target material, flux prediction, …
  - Some is important for context: Paper reference, …
- Comparison Inputs: Response matrices, background templates, flux uncertainties, …

What Is Needed to Predict a Cross Section Measurement?

- A Signal definition
- Event projection operators

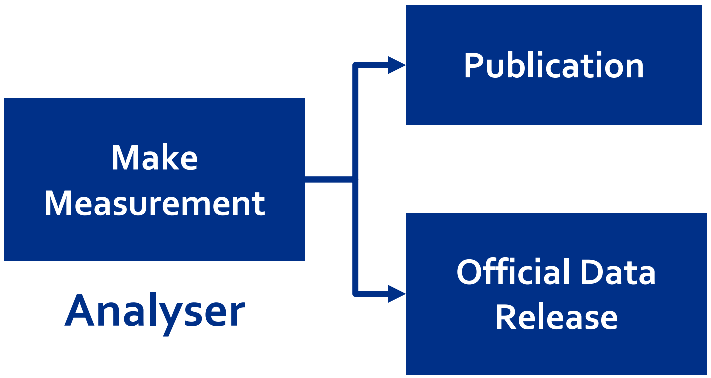
This bit is more ***difficult*** to modularise

- Measured data points
- Uncertainty descriptions
  - Errors on each bin
  - Covariance matrices across bins and measurements
  - Errors broken down by source
- Measurement metadata:
  - Some is important for making predictions: Target material, flux prediction, …
  - Some is important for context: Paper reference, …
- Comparison Inputs: Response matrices, background templates, flux uncertainties, …

These bits are mechanically ***easy*** to modularise, we just need to decide on a format

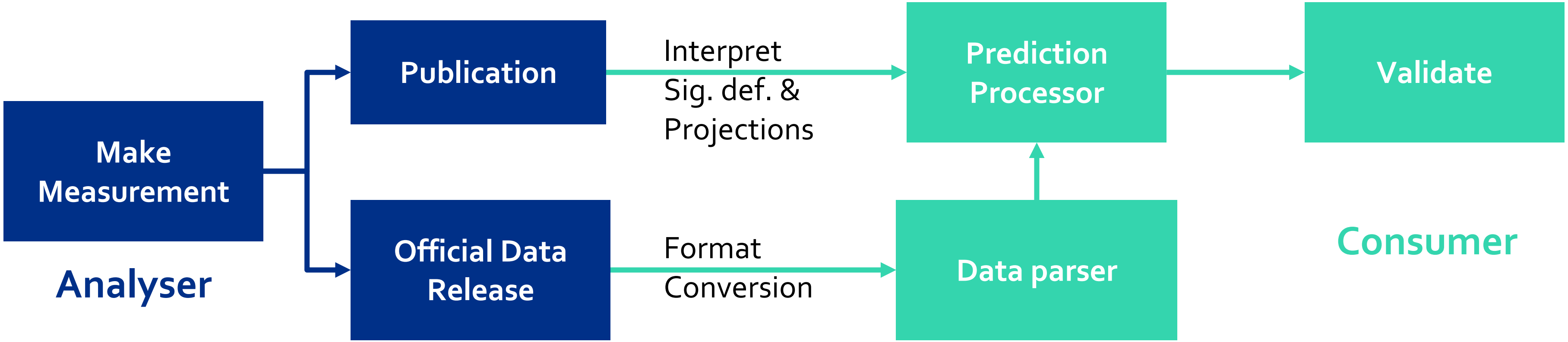
The Current Workflow

- Measurements are disseminated in publications and data releases



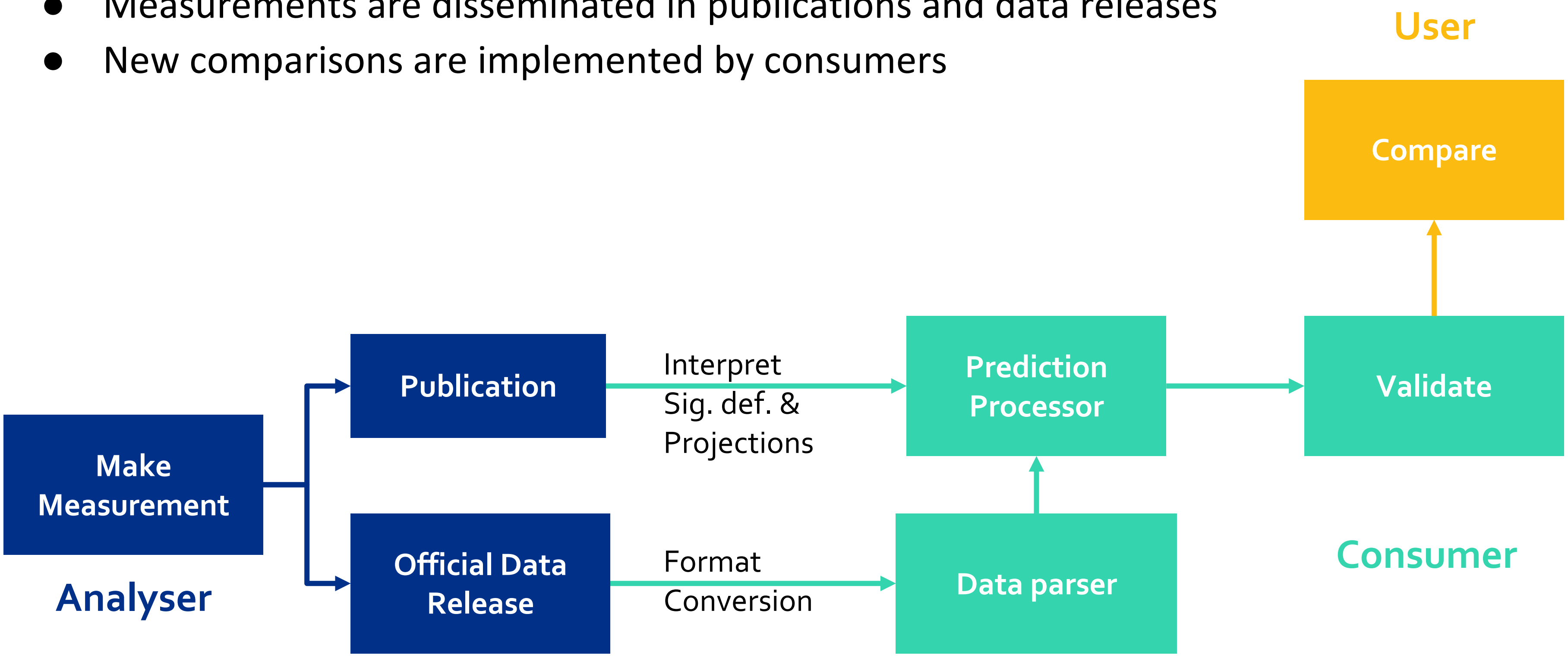
The Current Workflow

- Measurements are disseminated in publications and data releases
- New comparisons are implemented by consumers



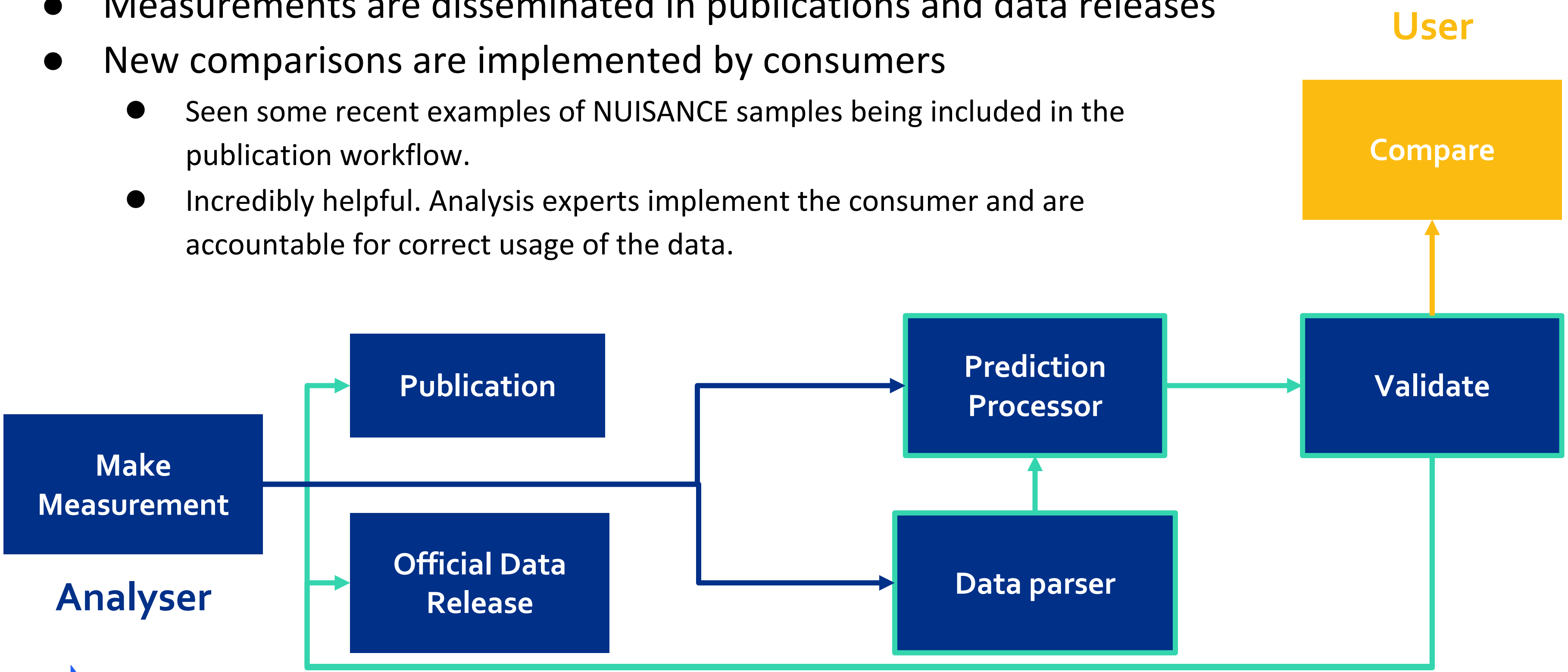
The Current Workflow

- Measurements are disseminated in publications and data releases
- New comparisons are implemented by consumers



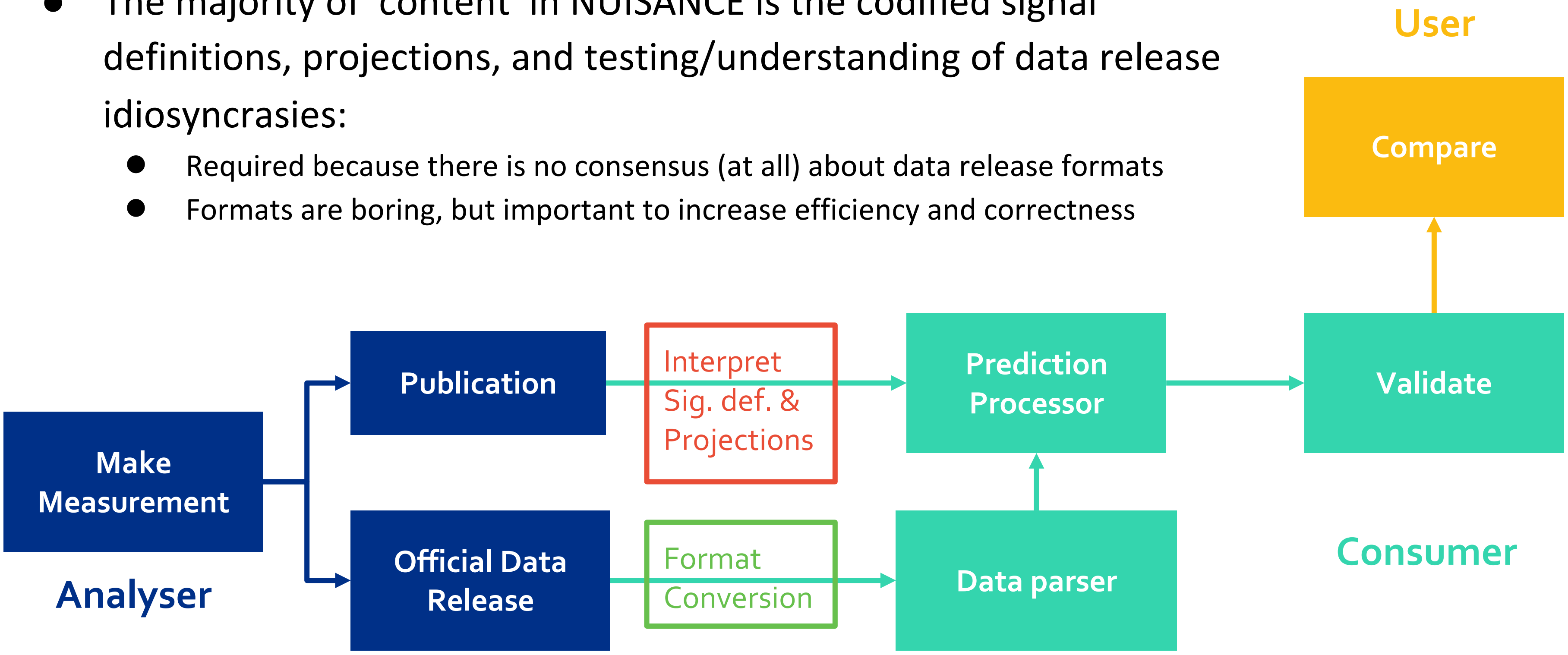
The Current Workflow

- Measurements are disseminated in publications and data releases
- New comparisons are implemented by consumers
  - Seen some recent examples of NUISANCE samples being included in the publication workflow.
  - Incredibly helpful. Analysis experts implement the consumer and are accountable for correct usage of the data.



The Current Workflow

- The majority of 'content' in NUISANCE is the codified signal definitions, projections, and testing/understanding of data release idiosyncrasies:
  - Required because there is no consensus (at all) about data release formats
  - Formats are boring, but important to increase efficiency and correctness



The Current Workflow

- The majority of 'content' in NUISANCE is the codified signal definitions, projections, and testing/understanding of data release idiosyncrasies:
  - Required because there is no **consensus** (at all) about data release formats
  - Forr

NuXTract 2023 - Towards a consensus in neutrino cross sections

User

Compare

Validate

Consumer

Make Measurement

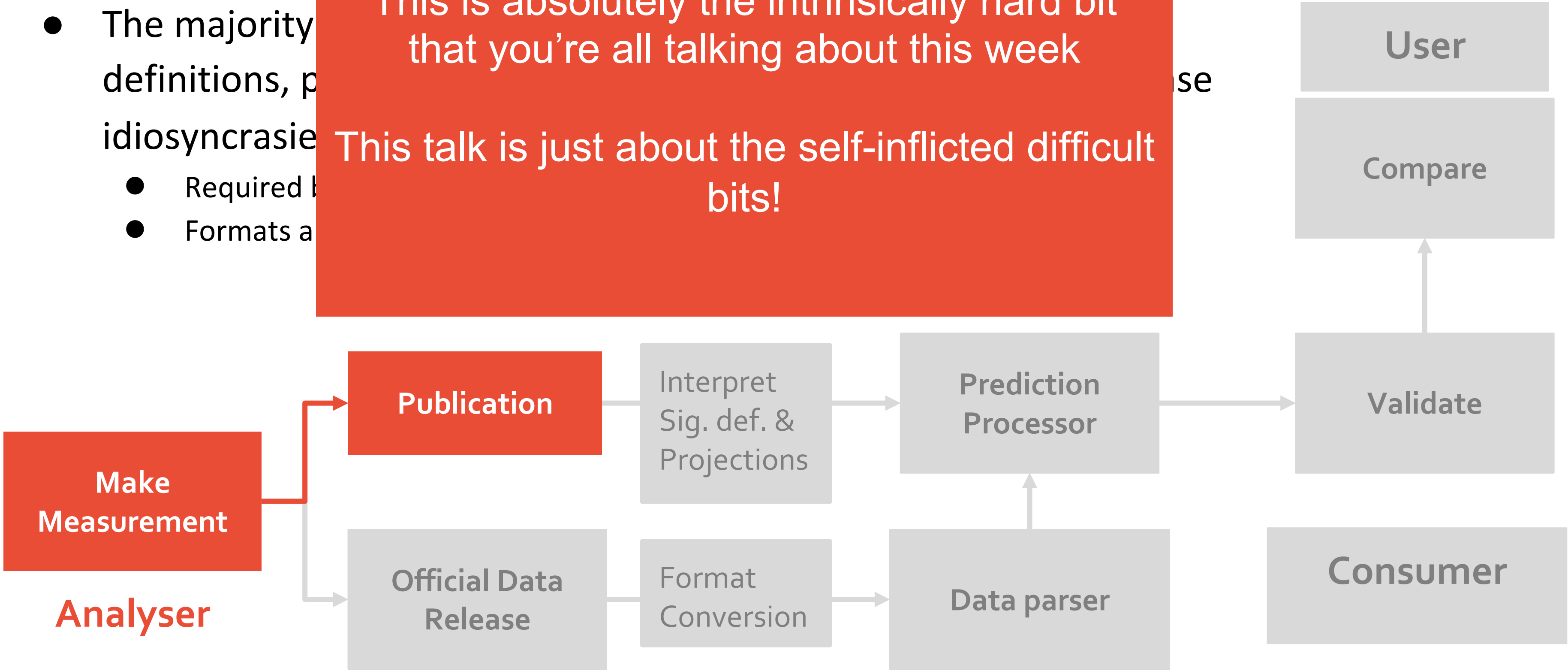
Analyser

The Curre

- The majority
  definitions, p … se
  idiosyncrasie
  - Required I
  - Formats a

This is absolutely the intrinsically hard bit that you're all talking about this week

This talk is just about the self-inflicted difficult bits!



What Should Be Improved

A number of problems for both the *analysers* and *consumers* in the existing model

What Should Be Improved

A number of problems for both the *analysers* and *consumers* in the existing model

Problems for analysers:

- Implementations are only useable in the full context of a specific consumer (NUISANCE)
- Bugs in consumers implementations can impact how many users consume data
    - NUISANCE is open source, bugs can propagate to other codebases
- Analysers are not getting any 'use' out of NUISANCE in terms of testing data releases or creating alternate generator predictions for publications
- Analysers wishing to contribute face an additional learning curve to implement and validate a new sample in NUISANCE

What Should Be Improved

A number of problems for both the *analysers* and *consumers* in the existing model

Problems for analysers:

- Implementations are only useable in the full context of a specific consumer (NUISANCE)
- Bugs in consumers implementations can impact how many users consume data
  - NUISANCE is open source, bugs can propagate to other codebases
- Analysers are not getting any 'use' out of NUISANCE in terms of testing data releases or creating alternate generator predictions for publications
- Analysers wishing to contribute face an additional learning curve to implement and validate a new sample in NUISANCE

Problems for consumers:

- Unclear who is accountable for the implementation:
  - Often not clear if data transformations were made to official releases, and if they were, were they correct
- No data-release standard formats mean most measurements require custom analysis code:
  - More effort to implement new measurements
  - Much larger code surface for bugs

What Should Be Improved

A number of problems for both the *analysers* and *consumers* in the existing model

Problems for analysers:

Presumption 1: Better documentation of individual result and worked examples for individual results is not the solution

When the expert producing them for Collab X moves on, we're back to square one as a community

additional learning curve to implement and validate a new sample in NUISANCE

What Should Be Improved

A number of problems for both the *analysers* and *consumers* in the existing model

Problems for analysers:

- :
- ere
- s
- additional learning curve to implement and validate a new sample in NUISANCE

Presumption 2: The solution is Open, Modular, and Standardised automated comparison tools which will build community knowledge and save everyone strife

Planning For Improvement

What Is Needed to Predict a Cross Section Measurement?

- A Signal definition
- Event projection operators

This bit is more ***difficult*** to modularise

- Measured data points
- Uncertainty descriptions
  - Errors on each bin
  - Covariance matrices across bins and measurements
  - Errors broken down by source
- Measurement metadata:
  - Some is important for making predictions: Target material, flux prediction, …
  - Some is important for context: Paper reference, …
- Comparison Inputs: Response matrices, background templates, flux uncertainties, …

These bits are mechanically ***easy*** to modularise, we just need to decide on a format

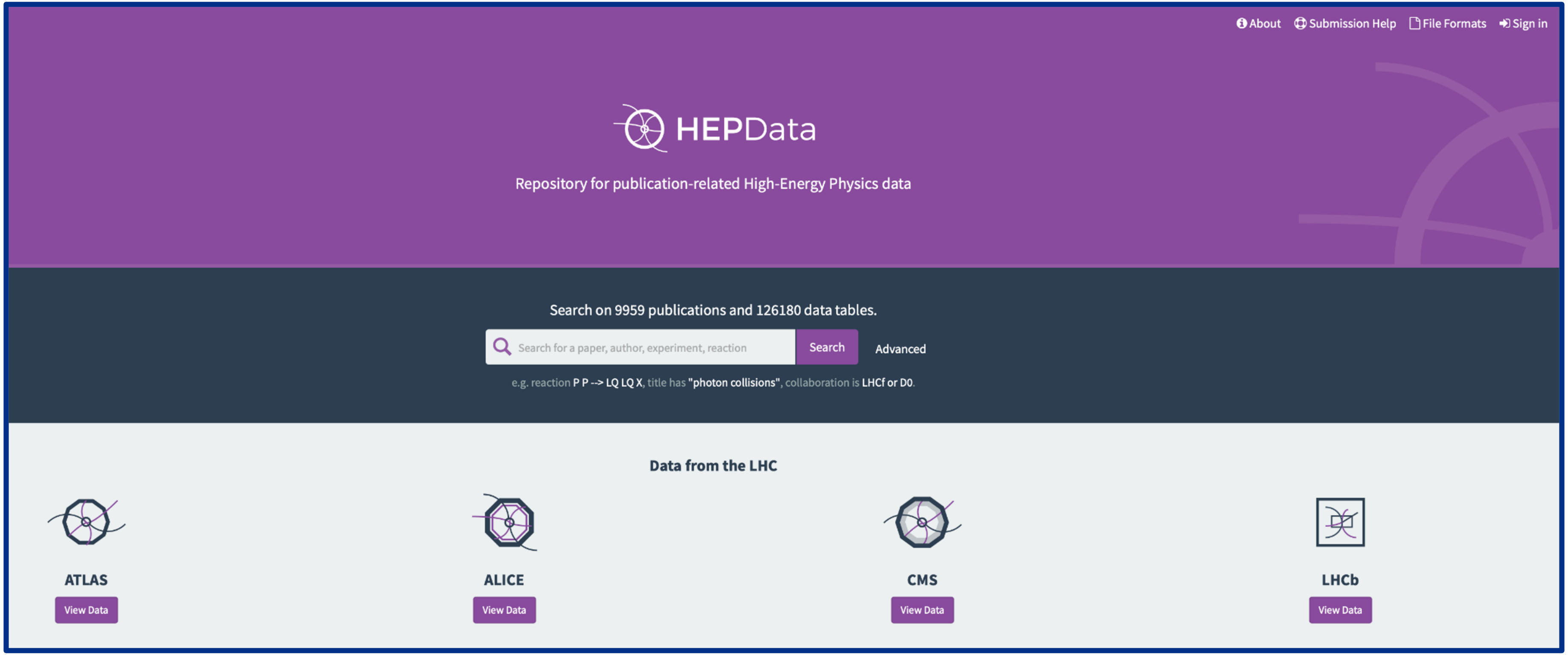
Looking Around For A Solution

- Deciding on a standard format for **the easy bit** seems like a good first step:
  - Need to make sure that it covers all our needs and is extensible into the foreseeable future
  - Ideally do no more work than is needed – Is there an existing standard that we can make use of?

Looking Around For A Solution

- Deciding on a standard format for **the easy bit** seems like a good first step:
  - Need to make sure that it covers all our needs and is extensible into the foreseeable future
  - Ideally do no more work than is needed – Is there an existing standard that we can make use of?
- We took a look over the shoulder of the Collider community to see what they were doing…

Looking Around For A Solution



HEPData

- A web repository of High Energy Physics Datasets:
  - Hosted at CERN, Funded by *UK Research and Innovation* (UKRI) via *The Institute for Particle Physics and Phenomenology* (IPPP) at Durham
  - Exposes RESTful API for programmatic retrieval of data.
- A Standard in the Collider measurement-comparison community:
  - RIVET, the equivalent tool to NUISANCE interfaces directly to HEPData

HEPData

- A web repository of High Energy Physics Datasets:
  - Hosted at CERN, Funded by *UK Research and Innovation* (UKRI) via *The Institute for Particle Physics and Phenomenology* (IPPP) at Durham
  - Exposes RESTful API for programmatic retrieval of data.
- A Standard in the Collider measurement-comparison community:
  - RIVET, the equivalent tool to NUISANCE interfaces directly to HEPData
- Defines a flexible data release format that has the extensibility to cover our needs
- Some datasets exist already:
  - Some built as demonstrations by IPPP staff to try and get us onboard in ~2017
  - MicroBooNE have put some of their releases up themselves

HEPData

HEPData | Search HEPData | Search

About | Submission Help | File Formats | Sign in

Browse all | Abratenko, P. et al. | Last updated on 2021-12-10 00:34 | Accessed 557 times | Cite | JSON

Hide Publication Information

First Measurement of Energy-dependent Inclusive Muon Neutrino Charged-Current Cross Sections on Argon with the MicroBooNE Detector

The MicroBooNE collaboration

Abratenko, P. , An, R. , Anthony, J. , Arellano, L. , Asaadi, J. , Ashkenazi, A. , Balasubramanian, S. , Baller, B. , Barnes, C. , Barr, G.

Phys.Rev.Lett. 128 (2022) 151801, 2022.

https://doi.org/10.17182/hepdata.114863

Journal | INSPIRE

Abstract (data abstract)
Data release for MicroBooNE's energy-dependent inclusive $\nu_m u$CC cross section result, corresponding to information from arXiv:2110.14023.

Download All

Filter 9 data tables

sigmaEnu
10.17182/hepdata.114863.v1/t1
ν_μCC inclusive total cross section per nucleon in each neutrino energy bin with statistical plus systematic uncertainty. The total uncertainty...

dsigmadEmu
10.17182/hepdata.114863.v1/t2
ν_μCC inclusive differential cross section per nucleon in each muon energy bin with statistical plus systematic uncertainty. The total uncertainty...

dsigmadnu
10.17182/hepdata.114863.v1/t3
ν_μCC inclusive differential cross section per nucleon in each hadronic energy transfer bin with statistical plus systematic uncertainty. The total...

cov_sigmaEnu
10.17182/hepdata.114863.v1/t4
Covariance matrix of the ν_μCC inclusive total cross section per nucleon in neutrino energy bins.

cov_dsigmadEmu
10.17182/hepdata.114863.v1/t5
Covariance matrix of the ν_μCC inclusive differential cross section per nucleon in muon energy bins.

cov_dsigmadnu

sigmaEnu 10.17182/hepdata.114863.v1/t1

https://www.hepdata.net/rec | JSON

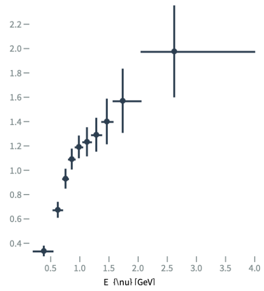
ν_μCC inclusive total cross section per nucleon in each neutrino energy bin with statistical plus systematic uncertainty. The total uncertainty comes from the square root of the covariance matrix diagonal entries.

phrases: neutrino-Ar interaction

reactions: inclusive numu CC

E_ν [GeV]	σ [$10^{-38} cm^2$/nucleon]
0.381800 (bin: 0.200000 - 0.540000)	0.336700 ±0.043690 stat+syst
0.622000 (bin: 0.540000 - 0.705000)	0.675100 ±0.065620 stat+syst
0.754600 (bin: 0.705000 - 0.805000)	0.930900 ±0.081930 stat+syst
0.861500 (bin: 0.805000 - 0.920000)	1.092000 ±0.086130 stat+syst
0.983300 (bin: 0.920000 - 1.050000)	1.192000 ±0.093540 stat+syst
1.122000 (bin: 1.050000 - 1.200000)	1.234000 ±0.119900 stat+syst
1.282000 (bin: 1.200000 - 1.375000)	1.292000 ±0.139400 stat+syst
1.463000 (bin: 1.375000 - 1.570000)	1.401000 ±0.187700 stat+syst
1.735000 (bin: 1.570000 - 2.050000)	1.571000 ±0.264100 stat+syst
2.619000 (bin: 2.050000 - 4.000000)	1.977000 ±0.378000 stat+syst

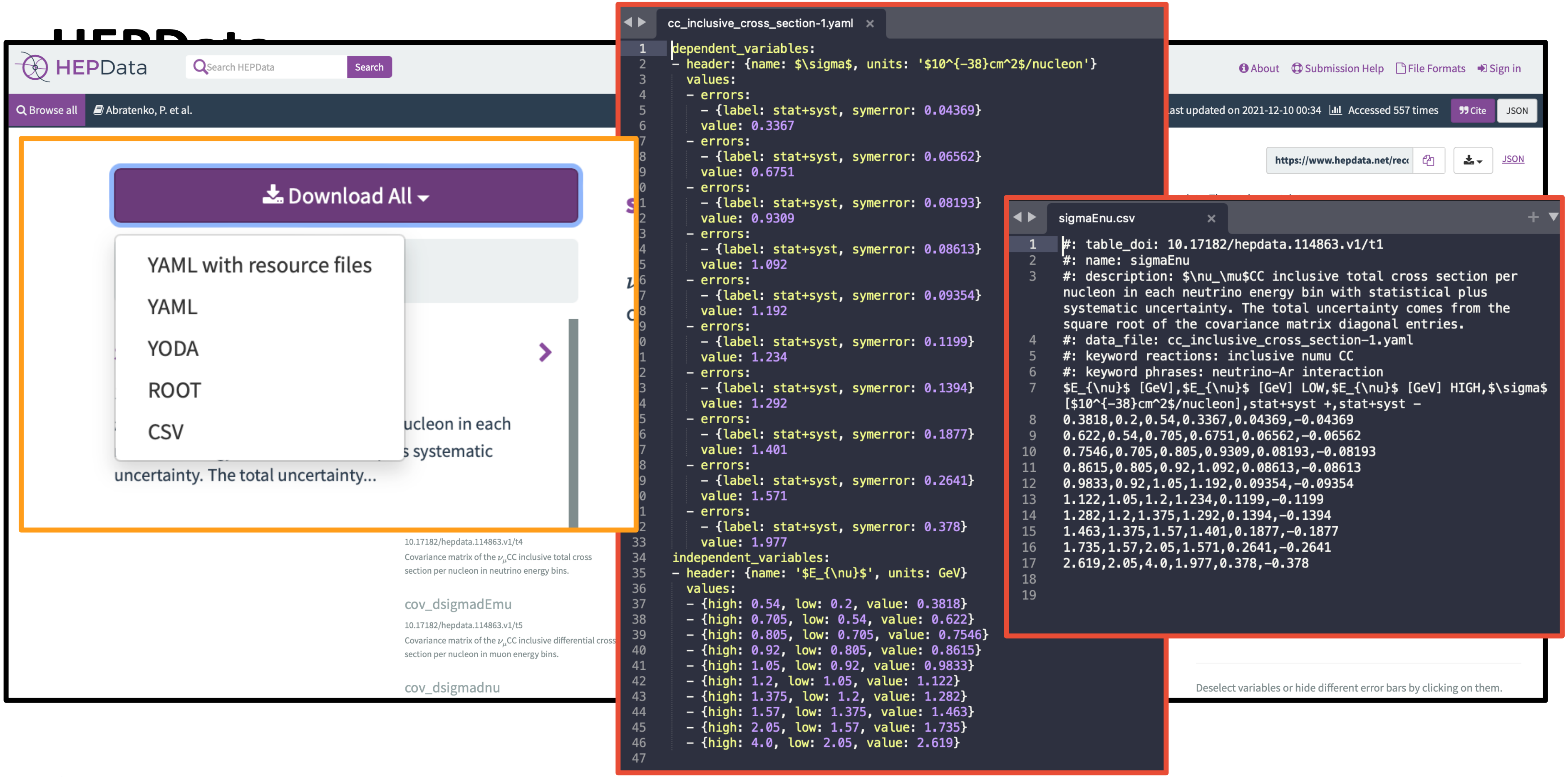
Visualize



Sum errors | Log Scale (X) | Log Scale (Y)

Deselect variables or hide different error bars by clicking on them.

HEPData



HEPData
Search HEPData
Search
About
Submission Help
File Formats
Sign in
Browse all
Abratenko, P. et al.
ast updated on 2021-12-10 00:34
Accessed 557 times
Cite
JSON
https://www.hepdata.net/rec
JSON
Download All
YAML with resource files
YAML
YODA
ROOT
CSV
ucleon in each
s systematic
uncertainty. The total uncertainty...
10.17182/hepdata.114863.v1/t4
Covariance matrix of the ν_μCC inclusive total cross section per nucleon in neutrino energy bins.
cov_dsigmadEmu
10.17182/hepdata.114863.v1/t5
Covariance matrix of the ν_μCC inclusive differential cross section per nucleon in muon energy bins.
cov_dsigmadnu
Deselect variables or hide different error bars by clicking on them.
cc_inclusive_cross_section-1.yaml
dependent_variables:
- header: {name: σ, units: '$10^{-38}cm^2$/nucleon'}
values:
- errors:
- {label: stat+syst, symerror: 0.04369}
value: 0.3367
- errors:
- {label: stat+syst, symerror: 0.06562}
value: 0.6751
- errors:
- {label: stat+syst, symerror: 0.08193}
value: 0.9309
- errors:
- {label: stat+syst, symerror: 0.08613}
value: 1.092
- errors:
- {label: stat+syst, symerror: 0.09354}
value: 1.192
- errors:
- {label: stat+syst, symerror: 0.1199}
value: 1.234
- errors:
- {label: stat+syst, symerror: 0.1394}
value: 1.292
- errors:
- {label: stat+syst, symerror: 0.1877}
value: 1.401
- errors:
- {label: stat+syst, symerror: 0.2641}
value: 1.571
- errors:
- {label: stat+syst, symerror: 0.378}
value: 1.977
independent_variables:
- header: {name: 'E_{ν}', units: GeV}
values:
- {high: 0.54, low: 0.2, value: 0.3818}
- {high: 0.705, low: 0.54, value: 0.622}
- {high: 0.805, low: 0.705, value: 0.7546}
- {high: 0.92, low: 0.805, value: 0.8615}
- {high: 1.05, low: 0.92, value: 0.9833}
- {high: 1.2, low: 1.05, value: 1.122}
- {high: 1.375, low: 1.2, value: 1.282}
- {high: 1.57, low: 1.375, value: 1.463}
- {high: 2.05, low: 1.57, value: 1.735}
- {high: 4.0, low: 2.05, value: 2.619}
sigmaEnu.csv
#: table_doi: 10.17182/hepdata.114863.v1/t1
#: name: sigmaEnu
#: description: ν_μCC inclusive total cross section per nucleon in each neutrino energy bin with statistical plus systematic uncertainty. The total uncertainty comes from the square root of the covariance matrix diagonal entries.
#: data_file: cc_inclusive_cross_section-1.yaml
#: keyword reactions: inclusive numu CC
#: keyword phrases: neutrino-Ar interaction
E_{ν} [GeV],E_{ν} [GeV] LOW,E_{ν} [GeV] HIGH,σ [$10^{-38}cm^2$/nucleon],stat+syst +,stat+syst -
0.3818,0.2,0.54,0.3367,0.04369,-0.04369
0.622,0.54,0.705,0.6751,0.06562,-0.06562
0.7546,0.705,0.805,0.9309,0.08193,-0.08193
0.8615,0.805,0.92,1.092,0.08613,-0.08613
0.9833,0.92,1.05,1.192,0.09354,-0.09354
1.122,1.05,1.2,1.234,0.1199,-0.1199
1.282,1.2,1.375,1.292,0.1394,-0.1394
1.463,1.375,1.57,1.401,0.1877,-0.1877
1.735,1.57,2.05,1.571,0.2641,-0.2641
2.619,2.05,4.0,1.977,0.378,-0.378

What Is Needed to Predict a Cross Section Measurement?

- A Signal definition
- Event projection operators

This bit is more ***difficult*** to modularise

- Measured data points ✅
- Uncertainty descriptions ✅
    - Errors on each bin
    - Covariance matrices across bins and measurements
    - Errors broken down by source
- Measurement metadata: ✅
    - Some is important for making predictions: Target material, flux prediction, …
    - Some is important for context: Paper reference, …
- Comparison Inputs: Response matrices, ✅ background templates, flux uncertainties, …

These bits are mechanically ***easy*** to modularise, we just need to decide on a format

HEPData

- A web repository of High Energy Physics Datasets:
  - Hosted at CERN, Funded by *UK Research and Innovation* (UKRI) via *The Institute for Particle Physics and Phenomenology* (IPPP) at Durham
  - Exposes RESTful API for programmatic retrieval of data.
- A Standard in the Collider measurement-comparison community:
  - RIVET, the equivalent tool to NUISANCE interfaces directly to HEPData
- Defines a flexible data release format that has the extensibility to cover our needs
- Some datasets exist already:
  - Some built as demonstrations by IPPP staff to try and get us onboard in ~2017
  - MicroBooNE have put some of their releases up themselves
- This is excellent as a standard, ASCII-first, extensible format
  - But does it enable automatic comparisons alone?

A HEPData Release

- The documentation for the HEPData data release submission format.
  - YAML-based with automatic conversion of data files to/from ROOT, JSON, YODA
  - Python tools maintained by contributors help prepare/manipulate HEPData records

A HEPData Release

- The documentation for the HEPData data release submission format.
  - YAML-based with automatic conversion of data files to/from ROOT, JSON, YODA
  - Python tools maintained by contributors help prepare/manipulate HEPData records
- Publications correspond to ***records*** and can be versioned similarly to arxiv records
- Sandbox allows controlled release testing and sharing before public listing
- Individual measurements correspond to ***tables***:
  - Can handle (un)binned in any number of dimensions with fully arbitrary hyper-rectangular bin definitions
  - Have space for arbitrary key/value metadata pairs called ***qualifiers***
  - ***Tables*** can also be used to store error matrices and flux predictions
- Arbitrary additional files can be included in the submission

A HEPData Release

- The documentation for the HEPData data release submission format.
  - YAML-based with automatic conversion of data files to/from ROOT, JSON, YODA
  - Python tools maintained by contributors help prepare/manipulate HEPData records
- Publications correspond to ***records*** and can be versioned similarly to arxiv records
- Sandbox allows controlled release testing and sharing before public listing
- Individual measurements correspond to ***tables***:
  - Can handle (un)binned in any number of dimensions with fully arbitrary hyper-rectangular bin definitions
  - Have space for arbitrary key/value metadata pairs called ***qualifiers***
  - ***Tables*** can also be used to store error matrices and flux predictions
- **Arbitrary additional files can be included in the submission**

The Easy

Helps with the Difficult

Standards For The Easy Bit

Proposing Standards: #1 Data Release

- Preparing a comprehensive, bespoke meta-data standard on top of the HEPData format that describes the minimum information required to predict a measurement:
  - HEPData
- Important for community use and preservation that these are defined separately from and one consumer framework (e.g. NUISANCE)

Proposing Standards: #1 Data Release

- Preparing a comprehensive, bespoke meta-data standard on top of the HEPData format that describes the minimum information required to predict a measurement:
  - HEPData
- Important for community use and preservation that these are defined separately from and one consumer framework (e.g. NUISANCE)
- At a high level, HEPData ***tables*** for automatic consumption must include:
  - An example implementation of the signal definition and kinematic projection operators
  - A reference to the recommended flux prediction for each neutrino species included in the signal
  - A description of the target material
  - For cross section measurements, be explicit about the cross section units
  - If a covariance matrix should be used, include a reference to the covariance matrix

Proposing Standards: #1 Data Release

- Preparing a comprehensive, bespoke meta-data standard on top of the HEPData format that describes the minimum information required to predict a measurement:
  - HEPData

- Important for community use and preservation that these are defined separately from and one consumer framework (e.g. NUISANCE)

- At a high level, HEPData ***tables*** for automatic consumption must include:
  - An example implementation of the signal definition and kinematic projection operators
  - A reference to the recommended flux prediction for each neutrino species included in the signal
  - A description of the target material
  - For cross section measurements, be explicit about the cross section units
  - If a covariance matrix should be used, include a reference to the covariance matrix

Proposing Standards: #1 Dat

- Preparing a comprehensive, bespoke me
  format that describes the minimum infor ... ent:
  - [HEPData]
- Important for community use and preser
  from and one consumer framework (e.g.
- At a high level, HEPData ***tables*** for auton
  - An example implementation of the signal defi
  - A reference to the recommended flux predicti
  - A description of the target material
  - For cross section measurements, be explicit a
  - If a covariance matrix should be used, include

Table of Contents

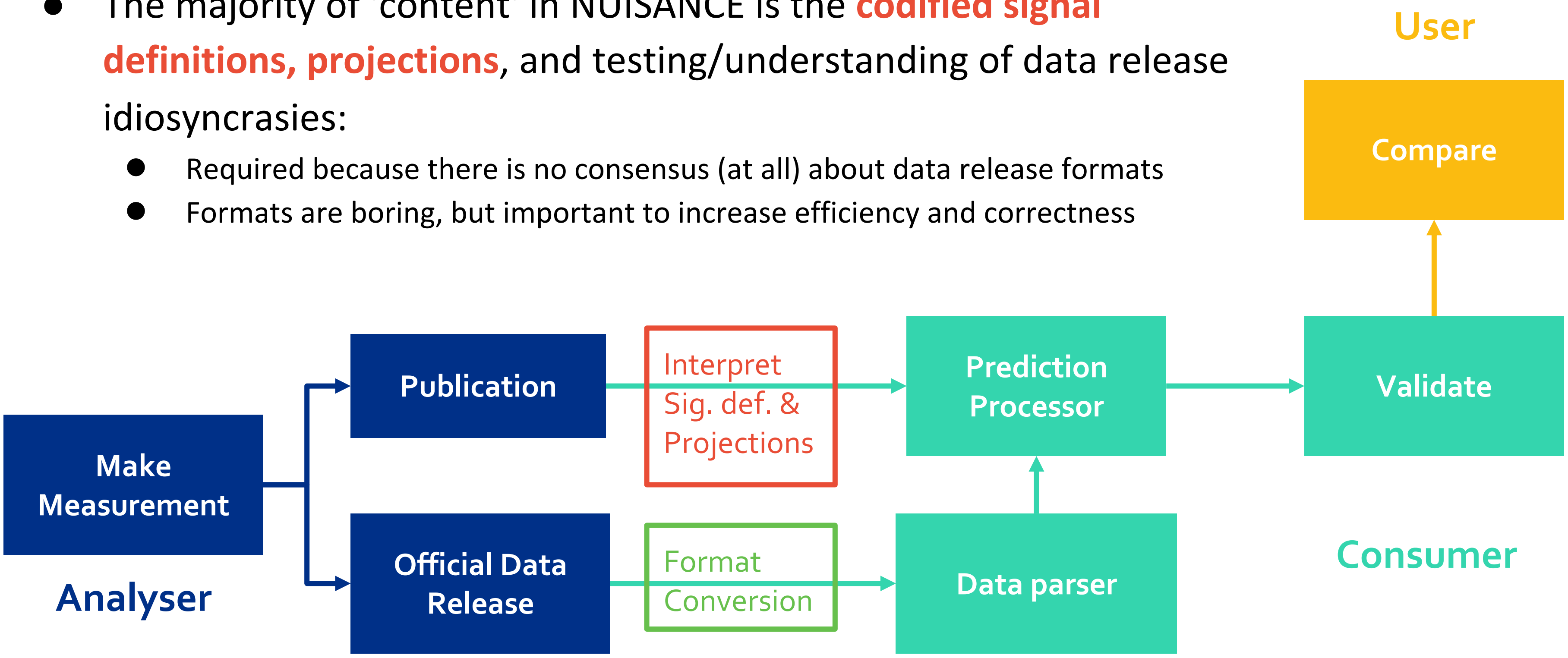
Considerations Not Discussed in Detail

- Multi-measurement correlations
- Flux predictions
- Multi-dimensional projections
- Inter-record references (See HepData:xxxyyy for the correct flux distribution)
- Multi-dimensional fluxes – Paving the way for PRISM (SBND, DUNE-, IWCD)
- Test-statistic recommendations/implementations
- Many other details you need to get right to not under/overconstrain possible measurements

Standards For The Difficult Bit

The Current Workflow

- The majority of 'content' in NUISANCE is the **codified signal definitions, projections**, and testing/understanding of data release idiosyncrasies:
  - Required because there is no consensus (at all) about data release formats
  - Formats are boring, but important to increase efficiency and correctness



Proposing Standards: #2 Event Processing Environment

- For comparisons with no custom code, we need a way of packaging executable signal definitions and projection operators with the data releases.
  - **Reminder:** Consumers (e.g. NUISANCE) already maintains all of this code, but it is currently distributed and *compiled* as part of NUISANCE

Proposing Standards: #2 Event Processing Environment

- For comparisons with no custom code, we need a way of packaging executable signal definitions and projection operators with the data releases.
    - **Reminder:** Consumers (e.g. NUISANCE) already maintains all of this code, but it is currently distributed and *compiled* as part of NUISANCE

- RIVET has a solution:
    - RIVET analyses can be expressed as free C++ functions and some boilerplate-generating macros
    - These are then often stored in HEPData along with the measurement and written by *analysers*
    - RIVET provides helper scripts for compiling these into dynamically-loadable plugins

Proposing Standards: #2 Event Processing Environment

- For comparisons with no custom code, we need a way of packaging executable signal definitions and projection operators with the data releases.
  - **Reminder:** Consumers (e.g. NUISANCE) already maintains all of this code, but it is currently distributed and *compiled* as part of NUISANCE

- RIVET has a solution:
  - RIVET analyses can be expressed as free C++ functions and some boilerplate-generating macros
  - These are then often stored in HEPData along with the measurement and written by *analysers*
  - RIVET provides helper scripts for compiling these into dynamically-loadable plugins

- We would like to reduce the number of helper scripts and user steps:
  - Use interpreters instead of compilers!

Proposing Standards: #2 Event Processing Environment

- For comparisons with no custom code, we need a way of packaging executable signal definitions and projection operators with the data releases.
  - **Reminder:** Consumers (e.g. NUISANCE) already maintains all of this code, but it is currently distributed and *compiled* as part of NUISANCE

- RIVET has a solution:
  - RIVET analyses can be expressed as free C++ functions and some boilerplate-generating macros
  - These are then often stored in HEPData along with the measurement and written by *analysers*
  - RIVET provides helper scripts for compiling these into dynamically-loadable plugins

- We would like to reduce the number of helper scripts and user steps:
  - Use interpreters instead of compilers!
  - But, being able to execute the functions is only one (mechanical) part of the problem

ProSelecta

- **Reminder:** The headline goal here is to enable automatic comparisons by removing the need for consumers to interpret and implement custom code per measurement

ProSelecta

- **Reminder:** The headline goal here is to enable automatic comparisons by removing the need for consumers to interpret and implement custom code per measurement
- **Proposal #2:** Define abstract event-processing utility functions in a language- and event-format- agnostic way.

ProSelecta

- **Reminder:** The headline goal here is to enable automatic comparisons by removing the need for consumers to interpret and implement custom code per measurement
- **Proposal #2:** Define abstract event-processing utility functions in a language- and event-format- agnostic way.

ProSelecta

Selections

```
GetBeam(event, PID) -> particle
GetBeamAny(event, list<PID>) -> particle

GetBeams(event, PID) -> list<particle>
GetBeamsAny(event, list<PID>) -> list<particles>

GetTarget(event) -> particle

GetOutPartFirst(event, PID) -> particle
GetOutPartFirstAny(event, list<PID>) -> particle

GetOutPartHM(event, PID) -> particle
GetOutPartHMAny(event, list<PID>) -> particle

GetOutParts(event, PID) -> list<particles>
GetOutPartsAny(event, list<PID>) -> list<particles>

GetOutPartsExcept(event, PID) -> list<particles>
GetOutPartsExceptAny(event, list<PID>) -> list<particles>
```

s to enable automatic comparisons by removing

t and implement custom code per measurement

-proce … d

```
parts::q0(particle, particle) -> real
parts::q3(particle, particle) -> real
parts::Q2Lep(particle, particle) -> real

parts::CosTheta(particle, particle) -> real
parts::Theta(particle, particle) -> real

parts::W(list<particles>) -> real
parts::EPmiss(list<particles>) -> 4vec
```

ProSelecta

- **Reminder:** The headline goal here is to enable automatic comparisons by removing the need for consumers to interpret and implement custom code per measurement
- **Proposal #2:** Define abstract event-processing utility functions in a language- and event-format- agnostic way.
- We are working on a *straw-person* standard and ref. implementation, ProSelectaCPP
  - Allows signal and projection functions to be written in C++ or python, completely equivalently

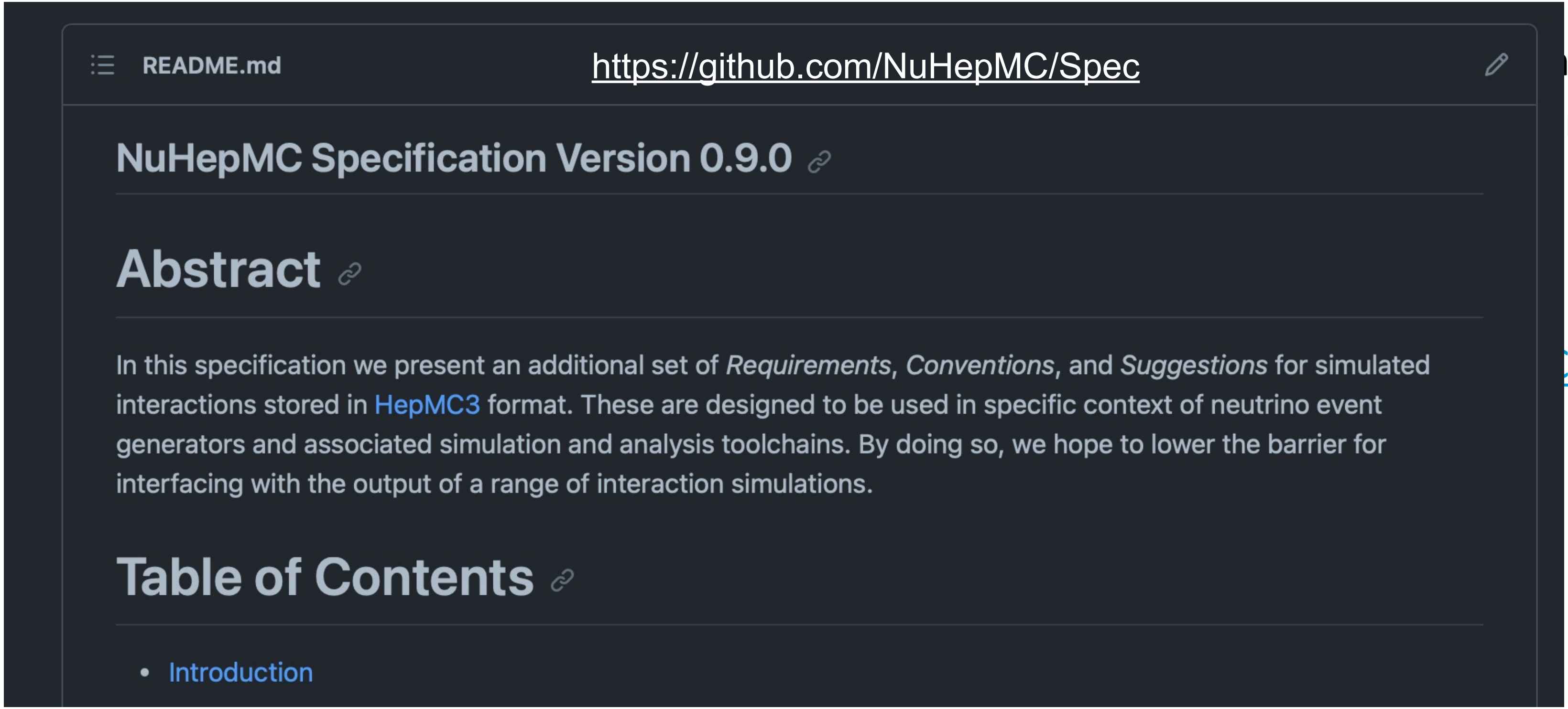
ProSelecta

- **Reminder:** The headline goal here is to enable automatic comparisons by removing the need for consumers to interpret and implement custom code per measurement
- **Proposal #2:** Define abstract event-processing utility functions in a language- and event-format- agnostic way.
- We are working on a *straw-person* standard and ref. implementation, ProSelectaCPP
  - Allows signal and projection functions to be written in C++ or python, completely equivalently

ProSelecta

- **Reminder:** The headline goal here is to enable automatic comparisons by removing the need for consumers to interpret and implement custom code per measurement
- **Proposal #2:** Define abstract event-processing utility functions in a language- and event-format- agnostic way.
- We are working on a *straw-person* standard and ref. implementation, ProSelectaCPP
  - Allows signal and projection functions to be written in C++ or python, completely equivalently
  - Operates on HepMC3 events, a collider standard
  - Extensible to other languages: dynamic or compiled if there is community interest

ProSelecta



https://github.com/NuHepMC/Spec

README.md

NuHepMC Specification Version 0.9.0

Abstract

In this specification we present an additional set of *Requirements*, *Conventions*, and *Suggestions* for simulated interactions stored in HepMC3 format. These are designed to be used in specific context of neutrino event generators and associated simulation and analysis toolchains. By doing so, we hope to lower the barrier for interfacing with the output of a range of interaction simulations.

Table of Contents

- Introduction

A MINERvA Example

A MINERvA Example

PRL.129.021803

README

Data release README. All files are contained in the associated compressed tarball - MINERvA_TripleDiffQELike_DataRelease.tar.gz

Signal definition-

PRL 129 021803

Kinematic windows:

P|| > 1.5 GeV
Muon Angle w.r.t neutrino < 20 degrees

Interaction and final state information:

Charge Current interaction of a muon type neutrino with the following particles allowed in the final state:
muon, any number of nucleons, gammas <10 MeV (nuclear deexcitation)

```
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

PRL.129.021803

README

```
Data release README. All files are contained in the associated compressed tarball - MINERvA_TripleDiffQELike_DataRelease.tar.gz

Signal definition-

Kinematic windows:

P|| > 1.5 GeV
Muon Angle w.r.t neutrino < 20 degrees


Interaction and final state information:

Charge Current interaction of a muon type neutrino with the following particles allowed in the final state:
muon, any number of nucleons, gammas <10 MeV (nuclear deexcitation)
```

PRL 129 021803

```
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

PRL.129.021803

README

```
Data release README. All files are contained in the associated compressed
tarball - MINERvA_TripleDiffQELike_DataRelease.tar.gz

Signal definition-

Kinematic windows:

P|| > 1.5 GeV
Muon Angle w.r.t neutrino < 20 degrees


Interaction and final state information:

Charge Current interaction of a muon type neutrino with the following
particles allowed in the final state:
muon, any number of nucleons, gammas <10 MeV (nuclear deexcitation)
```

PRL 129 021803

```cpp
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

PRL.129.021803

README

Data release README. All files are contained in the associated compressed tarball – MINERvA_TripleDiffQELike_DataRelease.tar.gz

Signal definition–

PRL 129 021803

Kinematic windows:

P|| > 1.5 GeV
Muon Angle w.r.t neutrino < 20 degrees

Interaction and final state information:

Charge Current interaction of a muon type neutrino with the following particles allowed in the final state:
muon, any number of nucleons, gammas <10 MeV (nuclear deexcitation)

```cpp
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

```cpp
double MINERvA_PRL129_021803_Project_MuonE(HepMC3::GenEvent &ev) {
  return ps::GetOutPartHM(ev, ps::pdg::kMuon)->E() * ps::GeV;
}

double MINERvA_PRL129_021803_Project_SumTp(HepMC3::GenEvent &ev) {
  double SumTP = 0;

  for (auto const &prot : ps::GetOutParts(ev, 2212)) {
    auto prot_4mom = prot->momentum();
    SumTP += (prot_4mom->e() - prot_4mom->m());
  }

  return SumTP * ps::GeV;
}
```

```cpp
double MINERvA_PRL129_021803_Project_q0QE(HepMC3::GenEvent &ev) {

  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);

  auto nu_4mom = nu->momentum();
  auto mu_4mom = mu->momentum();

  static double const m_mu = 105.66;
  static double const m_n = 939.565;
  static double const m_p = 938.272;
  static double const Eb = 34;

  static double const numer_const =
      std::pow(m_p, 2) - std::pow(m_n - Eb, 2) - std::pow(m_mu, 2);
  static double const denom_const = 2.0 * (m_n - Eb);

  double E_mu = mu_4mom->e();
  double p_mu = mu_4mom->p3mod();
  double cos_mu = ps::parts::CosTheta(nu, mu);
  double E_mu_less_p_mu_cos_mu = (E_mu - p_mu * cos_mu);

  double numer = numer_const + 2.0 * E_mu_less_p_mu_cos_mu * E_mu;
  double denom = denom_const - E_mu_less_p_mu_cos_mu;

  return (numer / denom) * ps::GeV;
}
```

```cpp
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

```
double MINERvA_PRL129_021803_Project_MuonE(HepMC3::GenEvent &ev) {
  return ps::GetOutPartHM(ev, ps::pdg::kMuon)->E() * ps::GeV;
}

double MINERvA_PRL129_021803_Project_SumTp(HepMC3::GenEvent &ev) {
  double SumTP = 0;

  for (auto const &prot : ps::GetOutParts(ev, 2212)) {
    auto prot_4mom = prot->momentum();
    SumTP += (prot_4mom->e() - prot_4mom->m());
  }

  return SumTP * ps::GeV;
}
```

```
double MINERvA_PRL129_021803_Project_q0QE(HepMC3::GenEvent &ev) {

  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);

  auto nu_4mom = nu->momentum();
  auto mu_4mom = mu->momentum();

  static double const m_mu = 105.66;
  static double const m_n = 939.565;
  static double const m_p = 938.272;
  static double const Eb = 34;

  static double const numer_const =
      std::pow(m_p, 2) - std::pow(m_n - Eb, 2) - std::pow(m_mu, 2);
  static double const denom_const = 2.0 * (m_n - Eb);

  double E_mu = mu_4mom->e();
  double p_mu = mu_4mom->p3mod();
  double cos_mu = ps::parts::CosTheta(nu, mu);
  double E_mu_less_p_mu_cos_mu = (E_mu - p_mu * cos_mu);

  double numer = numer_const + 2.0 * E_mu_less_p_mu_cos_mu * E_mu;
  double denom = denom_const - E_mu_less_p_mu_cos_mu;

  return (numer / denom) * ps::GeV;
}
```

```
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

```cpp
double MINERvA_PRL129_021803_Project_MuonE(HepMC3::GenEvent &ev) {
  return ps::GetOutPartHM(ev, ps::pdg::kMuon)->E() * ps::GeV;
}

double MINERvA_PRL129_021803_Project_SumTp(HepMC3::GenEvent &ev) {
  double SumTP = 0;

  for (auto const &prot : ps::GetOutParts(ev, 2212)) {
    auto prot_4mom = prot->momentum();
    SumTP += (prot_4mom->e() - prot_4mom->m());
  }

  return SumTP * ps::GeV;
}
```

```cpp
double MINERvA_PRL129_021803_Project_q0QE(HepMC3::GenEvent &ev) {

  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);

  auto nu_4mom = nu->momentum();
  auto mu_4mom = mu->momentum();

  static double const m_mu = 105.66;
  static double const m_n = 939.565;
  static double const m_p = 938.272;
  static double const Eb = 34;

  static double const numer_const =
      std::pow(m_p, 2) - std::pow(m_n - Eb, 2) - std::pow(m_mu, 2);
  static double const denom_const = 2.0 * (m_n - Eb);

  double E_mu = mu_4mom->e();
  double p_mu = mu_4mom->p3mod();
  double cos_mu = ps::parts::CosTheta(nu, mu);
  double E_mu_less_p_mu_cos_mu = (E_mu - p_mu * cos_mu);

  double numer = numer_const + 2.0 * E_mu_less_p_mu_cos_mu * E_mu;
  double denom = denom_const - E_mu_less_p_mu_cos_mu;

  return (numer / denom) * ps::GeV;
}
```

```cpp
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

```cpp
double MINERvA_PRL129_021803_Project_MuonE(HepMC3::GenEvent &ev) {
  return ps::GetOutPartHM(ev, ps::pdg::kMuon)->E() * ps::GeV;
}

double MINERvA_PRL129_021803_Project_SumTp(HepMC3::GenEvent &ev) {
  double SumTP = 0;

  for (auto const &prot : ps::GetOutParts(ev, 2212)) {
    auto prot_4mom = prot->momentum();
    SumTP += (prot_4mom->e() - prot_4mom->m());
  }

  return SumTP * ps::GeV;
}
```

```cpp
double MINERvA_PRL129_021803_Project_q0QE(HepMC3::GenEvent &ev) {

  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);

  auto nu_4mom = nu->momentum();
  auto mu_4mom = mu->momentum();

  static double const m_mu = 105.66;
  static double const m_n = 939.565;
  static double const m_p = 938.272;
  static double const Eb = 34;

  static double const numer_const =
      std::pow(m_p, 2) - std::pow(m_n - Eb, 2) - std::pow(m_mu, 2);
  static double const denom_const = 2.0 * (m_n - Eb);

  double E_mu = mu_4mom->e();
  double p_mu = mu_4mom->p3mod();
  double cos_mu = ps::parts::CosTheta(nu, mu);
  double E_mu_less_p_mu_cos_mu = (E_mu - p_mu * cos_mu);

  double numer = numer_const + 2.0 * E_mu_less_p_mu_cos_mu * E_mu;
  double denom = denom_const - E_mu_less_p_mu_cos_mu;

  return (numer / denom) * ps::GeV;
}
```

```cpp
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

```cpp
double MINERvA_PRL129_021803_Project_MuonE(HepMC3::GenEvent &ev) {
  return ps::GetOutPartHM(ev, ps::pdg::kMuon)->E() * ps::GeV;
}

double MINERvA_PRL129_021803_Project_SumTp(HepMC3::GenEvent &ev) {
  double SumTP = 0;

  for (auto const &prot : ps::GetOutParts(ev, 2212)) {
    auto prot_4mom = prot->momentum();
    SumTP += (prot_4mom->e() - prot_4mom->m());
  }

  return SumTP * ps::GeV;
}
```

```cpp
double MINERvA_PRL129_021803_Project_q0QE(HepMC3::GenEvent &ev) {

  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);

  auto nu_4mom = nu->momentum();
  auto mu_4mom = mu->momentum();

  static double const m_mu = 105.66;
  static double const m_n = 939.565;
  static double const m_p = 938.272;
  static double const Eb = 34;

  static double const numer_const =
      std::pow(m_p, 2) - std::pow(m_n - Eb, 2) - std::pow(m_mu, 2);
  static double const denom_const = 2.0 * (m_n - Eb);

  double E_mu = mu_4mom->e();
  double p_mu = mu_4mom->p3mod();
  double cos_mu = ps::parts::CosTheta(nu, mu);
  double E_mu_less_p_mu_cos_mu = (E_mu - p_mu * cos_mu);

  double numer = numer_const + 2.0 * E_mu_less_p_mu_cos_mu * E_mu;
  double denom = denom_const - E_mu_less_p_mu_cos_mu;

  return (numer / denom) * ps::GeV;
}
```

```cpp
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

- The aim is that this environment will allow very declarative signal and projection functions to be written and packaged with the measurements

```cpp
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

- The aim is that this environment will allow very declarative signal and projection functions to be written and packaged with the measurements
- In NUISANCE would use `ProSelecta` to run this actual code on MC events to select and project them

```cpp
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

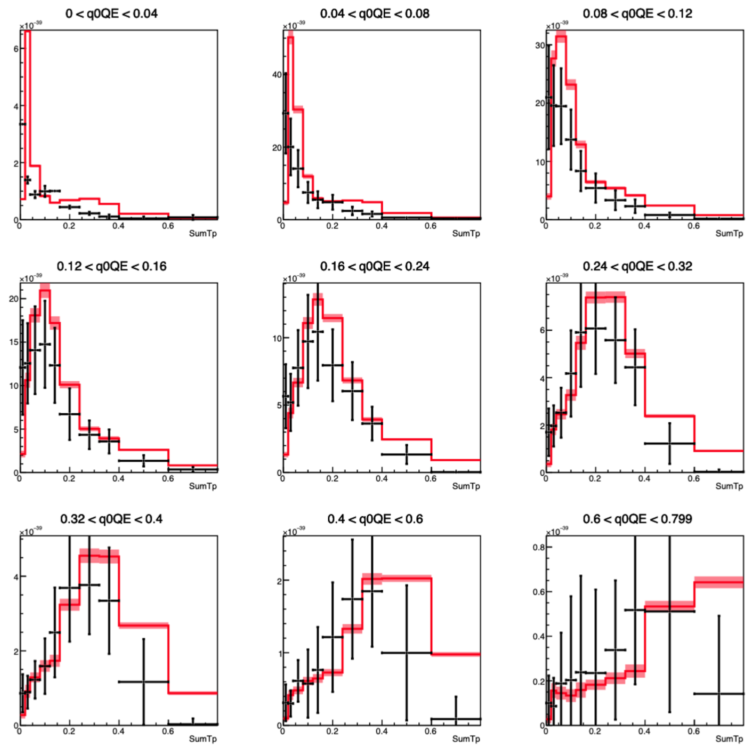
A MINERvA Example

- The aim is that this environment will allow very declarative signal and projection functions to be written and packaged with the measurements
- In NUISANCE would use `ProSelecta` to run this actual code on MC events to select and project them
- Other frameworks can also use `ProSelecta` or their own implementation or can read the code and transcribe it to their own framework

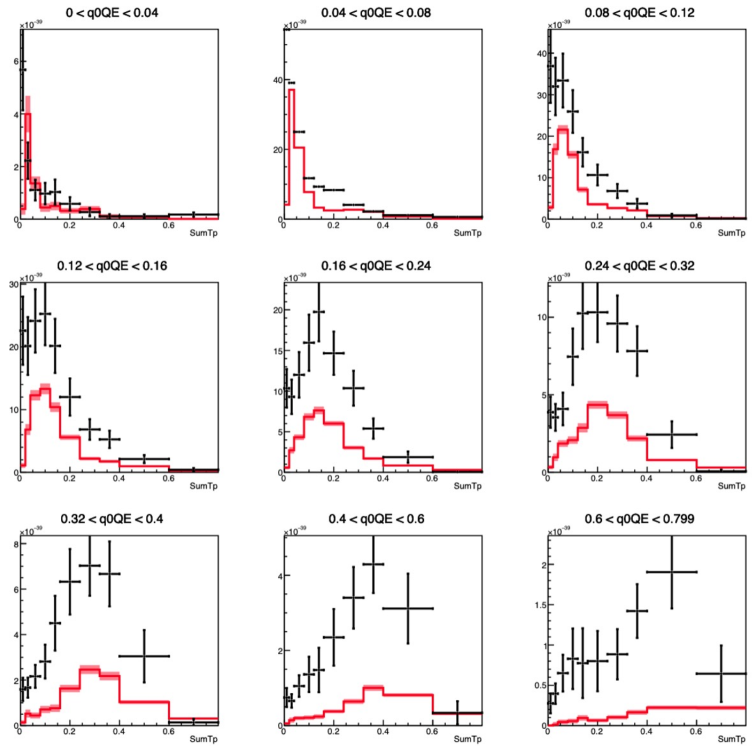
```cpp
bool MINERvA_PRL129_021803_SignalDefinition(HepMC3::GenEvent const &ev) {
  auto nu = ps::GetBeam(ev, ps::pdg::kNuMu);
  auto mu = ps::GetOutPartHM(ev, ps::pdg::kMuon);
  if (!nu || !mu) {
    return false;
  }

  auto ct = ps::parts::CosTheta(nu, mu);

  // check muon angle
  if ((std::acos(ct) * ps::deg) >= 20.0) {
    return false;
  }

  // check p||
  if ((ct * mu->momentum().length() * ps::GeV) < 1.5) {
    return false;
  }

  // Check no gammas above 10 MeV
  auto gammas = ps::GetOutParts(ev, ps::pdg::kGamma);
  for (auto &g : gammas) {
    if ((g->momentum().e() * ps::MeV) >= 10) {
      return false;
    }
  }

  auto other_parts = ps::GetOutPartsExceptAny(
      ev, {ps::pdg::kNuMu, ps::pdg::kMuon, ps::pdg::kGamma, ps::pdg::kProton,
           ps::pdg::kNeutron});

  return (other_parts.size() == 0);
}
```

A MINERvA Example

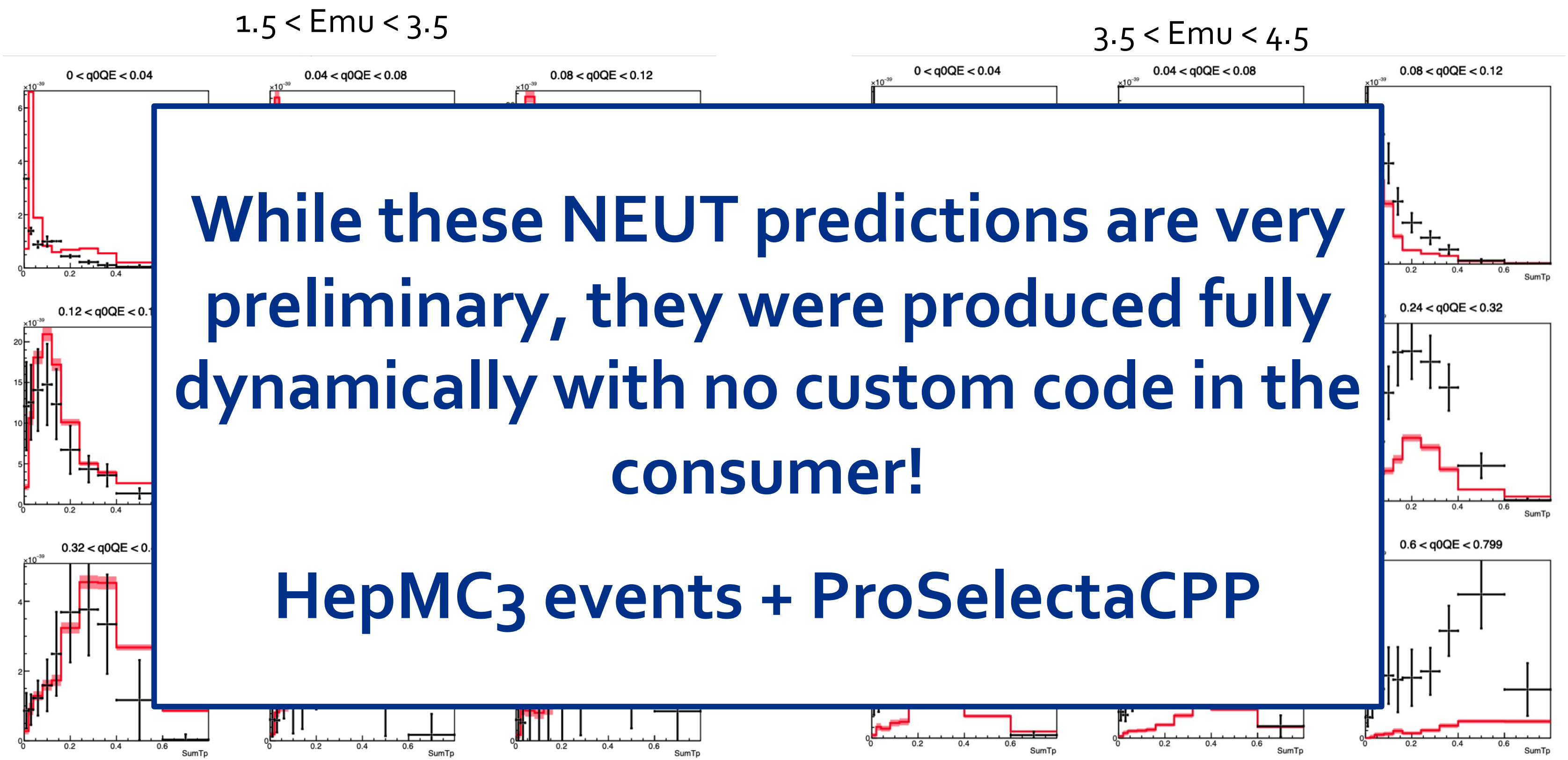
1.5 < Emu < 3.5



3.5 < Emu < 4.5



A MINERvA Example



While these NEUT predictions are very preliminary, they were produced fully dynamically with no custom code in the consumer!

HepMC3 events + ProSelectaCPP

Considerations Not Discussed in Detail

- Who decides what functions enter the standard?
  - Need to take care when standardizing composite variables:
    - Is MINERvA's Q2QE the same as MicroBooNE's?
    - Best to have analyser-written projections for each measurement
- Theorists don't use ROOT, extra tools sounds like extra overhead, not less!
  - 'Provide things in ASCII' will never be standard enough to not introduce space for bugs and miscommunication and misinterpretation, no matter the detail of documentation.
  - Is pure python better?
- Standard function set should be small and simple enough to be easily transcribable to a different framework.
  - The implementations are meant as example implementations, we just provide tools to execute them directly if you so wish. Aids analyser validations before publication

Considerations Not Discussed in Detail

- Who decides what functions enter the standard?
    - Ne
- Theori
    - ‘P
      m
    - Is
- Standa ... ribable
  to a di
    - Th ... ute them
      di

None of this is an excuse for less documentation in papers and data release supplementary materials.

We need both, clear documentation, and analyser-vetted implementations

Summary

The Proposed Workflow

The Proposed Workflow

- This is information that *analysers* have anyway, we are just trying to provide stable community tooling to allow them to share and preserve that:
    - In return, the barrier for analysers to test their data release and make predictions from a range of simulations for their publication precipitously drops

The Proposed Workflow

- This is information that *analysers* have anyway, we are just trying to provide stable community tooling to allow them to share and preserve that:
  - In return, the barrier for analysers to test their data release and make predictions from a range of simulations for their publication precipitously drops
- On the *consumer* side, we will know that the implementation is vetted by the experiment and is stored in a standardised way that is independent of our framework
  - Plots equivalent to those in the paper producible 'for free'

The Proposed Workflow

- This is ... stable comm...
  - In ... nge of si...
- On the ... e experi... amework

Make Measuremen...

Analyser

If you are interested in anything I have said, please get in touch.

Plan:

-- Finish and disseminate first draft on arXiv
-- Hold a dedicated workshop Summer 2024

In Summary

- We think that this proposal is a win-win:
  - Clarifies responsibility and 'ownership' for data releases that people use
    - HEPData links to journals and INSPIRE automatically
  - Standardise things that can be fully standardised and provides tools for everything else
  - Dramatically reduces maintenance load for consumer frameworks without loss of functionality

In Summary

- We think that this proposal is a win-win:
  - Clarifies responsibility and 'ownership' for data releases that people use
    - HEPData links to journals and INSPIRE automatically
  - Standardise things that can be fully standardised and provides tools for everything else
  - Dramatically reduces maintenance load for consumer frameworks without loss of functionality

- This has to be a community project, we are presenting one starting point for discussion:
  - If we design and promote it together, then it will help for future measurements

- The aim is to save everyone time

In Summary

- We think that this proposal is a win-win:
  - Clarifies responsibility and 'ownership' for data releases that people use
    - HEPData links to journals and INSPIRE automatically
  - Standardise things that can be fully standardised and provides tools for everything else
  - Dramatically reduces maintenance load for consumer frameworks without loss of functionality

- This has to be a community project, we are presenting one starting point for discussion:
  - If we design and promote it together, then it will help for future measurements

- The aim is to save everyone time (but mainly me).

In Summary

- We think that this proposal is a win-win:
  - Clarifies responsibility and 'ownership' for data releases that people use
    - HEPData links to journals and INSPIRE automatically
  - Standardise things that can be fully standardised and provides tools for everything else
  - Dramatically reduces maintenance load for consumer frameworks without loss of functionality
- This has to be a community project, we are presenting one starting point for discussion:
  - If we design and promote it together, then it will help for future measurements
- The aim is to save everyone time (but mainly me).

Backup

ProSelecta

- Again start with a standard: https://github.com/NUISANCEMC/ProSelecta
- Define abstract utility functions in a language- and event-format-agnostic way and then provide useful reference implementations in languages and frameworks that we want to use.
- Minimal type system

Types

The ProSelecta type system is defined below:

- bool
- real
- PID: An integer identifier that specifies particle species. See PDG 2023.
- 4vec: A 4-vector
- particle
- event
- list<T>: A generic container of a single, specified type, *e.g.* list<particle>.

ProSelecta

- Again start with a standard: https://github.com/NUISANCEMC/ProSelecta
- Define abstract utility functions in a language- and event-format-agnostic way and then provide useful reference implementations in languages and frameworks that we want to use.
- Minimal type system
- Language-agnostic function prototypes

```
qual::MyFunction(event) -> list<particle>
```

which describes a function, `MyFunction`, in namespace or module or with prefix `qual`, which takes an `event` as input and returns a `list` of `particle`s. Depending on the language the actual function invocation may look like: `qual::MyFunction`, `qual.MyFunction`, or `qual_MyFunction`, see documentation for the concrete implementation for explicit details.

ProSelecta: Particle Selection Utilities

- The intended inputs and outputs of these particle selection functions will be fully described by the standard

Selections

```
GetBeam(event, PID) -> particle
GetBeamAny(event, list<PID>) -> particle

GetBeams(event, PID) -> list<particle>
GetBeamsAny(event, list<PID>) -> list<particles>

GetTarget(event) -> particle

GetOutPartFirst(event, PID) -> particle
GetOutPartFirstAny(event, list<PID>) -> particle

GetOutPartHM(event, PID) -> particle
GetOutPartHMAny(event, list<PID>) -> particle

GetOutParts(event, PID) -> list<particles>
GetOutPartsAny(event, list<PID>) -> list<particles>

GetOutPartsExcept(event, PID) -> list<particles>
GetOutPartsExceptAny(event, list<PID>) -> list<particles>
```

ProSelecta: Particle Kinematic Projections

- The mathematical form of each kinematic projection will be fully, unambiguously defined by the standard:
  - Implementers can decide if an existing function is the one they need for their analysis, or if they need to implement their kinematic projection from the particle stack
  - We don't want to be fully comprehensive with the provided utility functions, but might appreciate thoughtful input on what projections to standardise:
    - e.g. EAvail? I expect there are enough different definitions of some similar quantity, that it might be difficult.

```
parts::q0(particle, particle) -> real
parts::q3(particle, particle) -> real
parts::Q2Lep(particle, particle) -> real

parts::CosTheta(particle, particle) -> real
parts::Theta(particle, particle) -> real

parts::W(list<particles>) -> real
parts::EPmiss(list<particles>) -> 4vec
```

ProSelecta: Hard-scatter Channel Selections

- To be able to describe some published neutrino-scattering measurements we need to be able to make signal definitions based on true hard-scatter channel

```
IsCC(event) -> bool
IsCOH(event) -> bool
Is1p1h(event) -> bool
Is2p2h(event) -> bool
IsSPP(event) -> bool
IsRES(event) -> bool
IsDIS(event) -> bool
```