PyHF use-case:
Branching Ratio
Determination of

\[ B \rightarrow D^{**} \ell \nu \]

Noreen Rauls
(Universität Göttingen)
Analysis Goal

- measurement of branching ratio $B \rightarrow D^{**} \ell \nu$ with hadronic FEI
- normalised to $B \rightarrow D^* \ell \nu$
- in total two fits
Analysis Overview

Decay channel

Neutral

\[ B^0 \rightarrow D^- \pi^0 \ell^+ \]
\[ B^0 \rightarrow D^{*-} \pi^0 \ell^+ \]
\[ B^0 \rightarrow D^0 \pi^- \ell^+ \]
\[ B^0 \rightarrow \bar{D}^0 \pi^- \ell^+ \]

Charged

\[ B^- \rightarrow D^0 \pi^0 \ell^- \]
\[ B^- \rightarrow D^{*-0} \pi^0 \ell^- \]
\[ B^- \rightarrow D^+ \pi^- \ell^- \]
\[ B^- \rightarrow D^{*+} \pi^- \ell^- \]

Neutral and charged $B_{S\pi}$ decay channels

\[ B \rightarrow D^{**} \ell \nu \]

Noreen Rauls
General Fit Setup

- do binned maximum likelihood fit in
  with bins in range

- combine backgrounds due to similar shapes
- divide signal into individual decay channels
General Fit Setup

- combine backgrounds due to similar shapes
  → 8 signal categories for D**
  → 3 background categories
General Fit Setup

- to incorporate between overlap between $D^{**}$ reconstruction modes
  → one overall fit for each lepton mode
  → e and mu

$B \rightarrow D^{**} \ell \nu$
General Fit Setup

- include isospin constraints

- Same scaling parameter

```
"modifiers": [
  {
    "name": "mu",
    "type": "normfactor",
    "data": null
  }
]
```
Systematic Uncertainties

- included systematic uncertainties as nuisance parameters
  - LeptonID, pi0, HadronID, Tracking, slow pi, photon, lumi

- for lumi include as parameter

- Statistical uncertainties from MC

```
"parameters": [ 
  { 
    "name": "lumi",
    "auxdata": [1.0],
    "sigmas": [0.017],
    "bounds": [[[0.915, 1.085]],
    "inits": [1.0],
  },
],
```

```
"modifiers": [ 
  { 
    "name": "my_staterror",
    "type": "staterror",
    "data": [1.0, 2.0],
  }
],
```
Systematic Uncertainties

- per-bin shape uncertainty controlled by single nuisance parameter (gaussian constraint)

→ in Belle II: most systematic uncertainties consist of stat. and syst. component
My Model

- Dictionary writing for model automated → many entries

\[ B \rightarrow D^{*\ast} \ell \nu \]
Running the Fit

- possible to use minuit or scipy for minimising the likelihood
  → only minuit returns uncertainties

- Use scipy as initial guess
  → fit twice
  → fit converges quicker
Results with Asimov Data

- Sample ordering might have changed
Results with Asimov Data

- get results after the fits
  → interpolation done automatically for nuisance parameters
  → returns values per bin

\[ B \rightarrow D^{**}\nu \]
Get Errors using 

\[ B \rightarrow D^{**} \ell \nu \]

- for fit uncertainties:
- subtract statistical error from fit uncertainties in quadrature
  \[ \rightarrow \text{one fit with scaling parameters only (statistical uncertainty)} \]
  \[ \rightarrow \text{then fit with everything} \]
Pull distributions

- investigate pull distributions
  → code taken from this example
- possible to change label names
Toy Study

```python
def generate_toys(
    model: pyhf.pdf.Model,
    random_seed: int,
    n_toys: int = 100,
) -> np.ndarray:
    np.random.seed(random_seed)
    toys = model.make_pdf(
        pyhf.tensorlib.astensor(np.ndarray(model.config.suggested_init()))
    ).sample((n_toys,))
    return toys
```

- Use histogram and Poisson smear every bin → looks more data like

- Histogram before the fit

\[ B \rightarrow D^{**} \ell \bar{\nu} \]
Toy Study

```python
def generate_toys(
    model: pyhf.pdf.Model,
    random_seed: int,
    n_toys: int = 500,
) -> np.ndarray:
    """
    Return a Tuple with a n_toys performed for a certain model.
    """
    np.random.seed(random_seed)
    toys = model.make_pdf(
        pyhf.tensorlib.astensor(np.asarray(model.config.suggested_init()))
    ).sample((n_toys,))
    return toys
```

- Use histogram and Poisson smear every bin
  → looks more data like

- Histogram after the fit

$B \rightarrow D^{**} \ell \nu$
Systematic Uncertainties

- possible to build workspace and remove particular modifier
  → use to determine effect of systematic uncertainties

new_workspace = workspace.prune(
    modifiers=not_considered_modifiers
)

- fit with all systematic uncertainties
- remove particular systematic uncertainty to determine impact on overall result
  → also automated that
Conclusion

- PyHF works well to determine branching ratio of $B \rightarrow D^{**}\ell\nu$
- examples good to get a first understanding
  → more complex example for binned HEP analysis without hypotesting would be helpful
- need to determine bin counts and uncertainties yourself
  → has been automated for this analysis