

Overview of Systematics framework

[Repository Documentation](#)

KEKCC: /group/belle2/dataproduct/Systematics/

Sviatoslav BILOKIN
LMU München

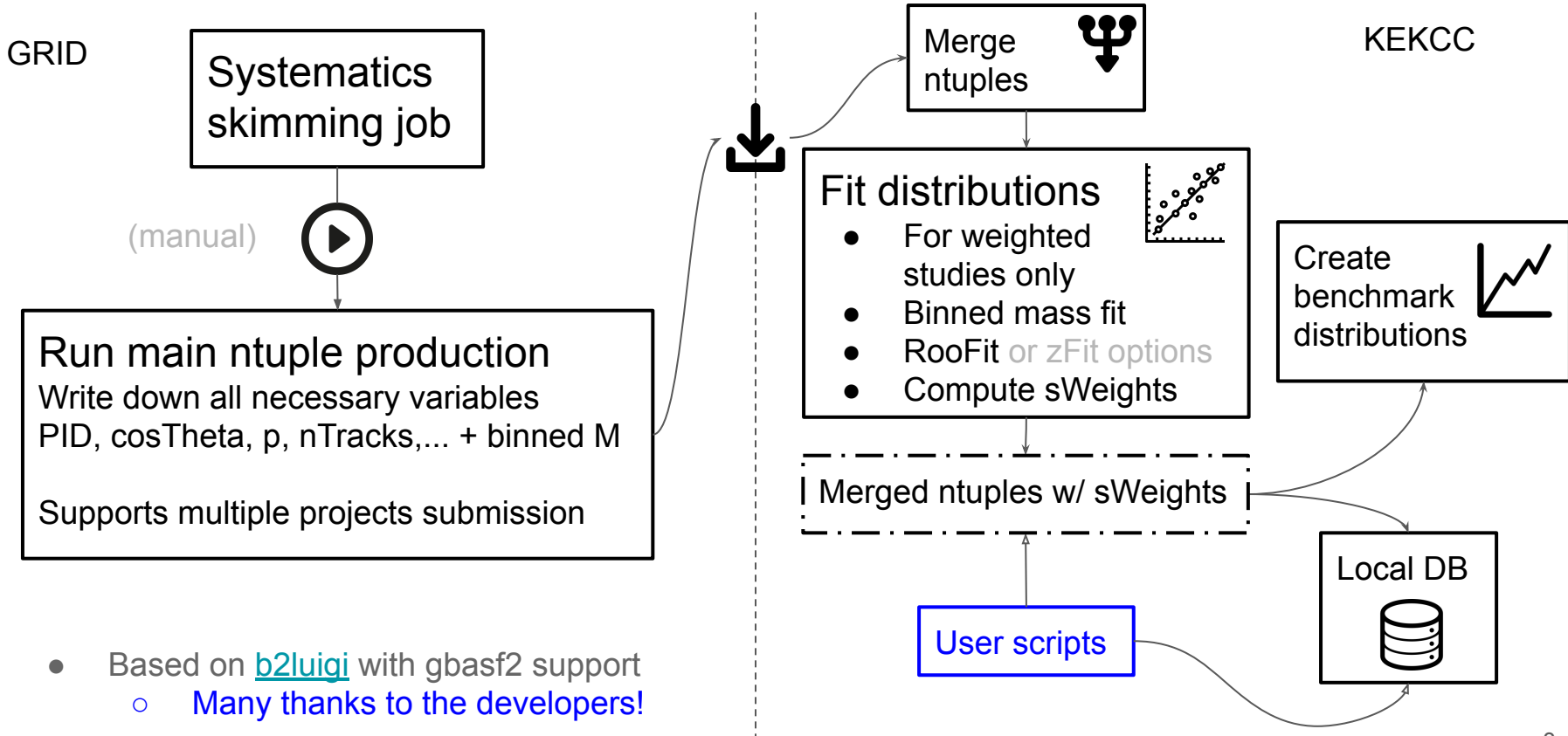


Introduction

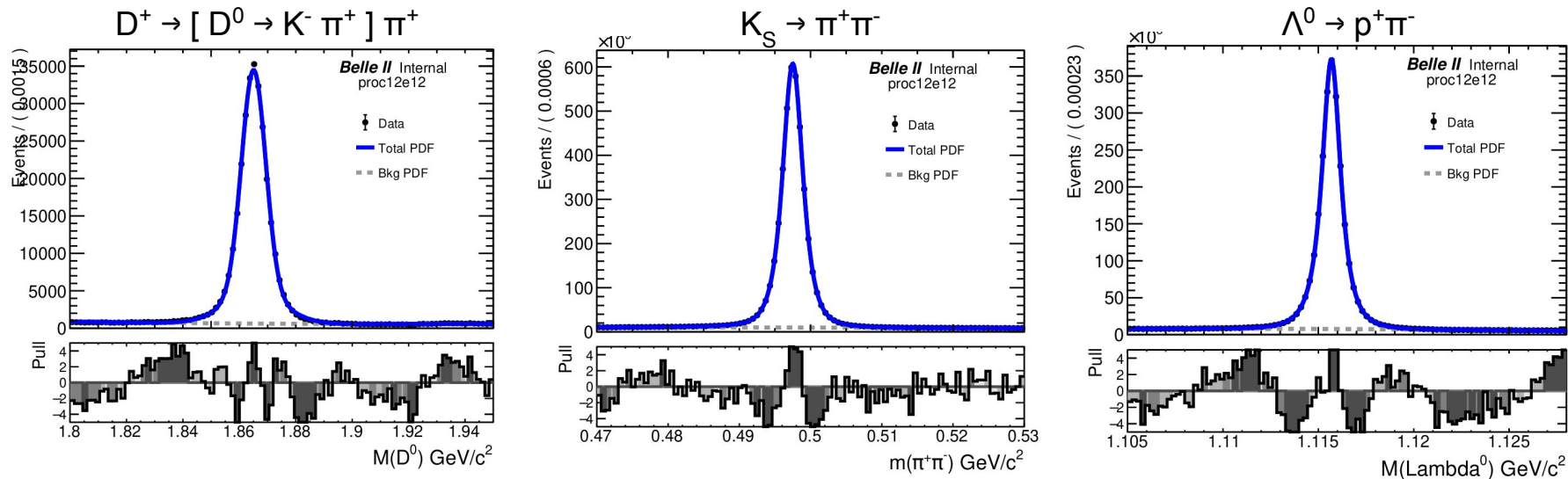
- We are working on a framework that should unify and automate computation of systematic uncertainties
 - Automation is needed since the production of correction weights is a repetitive task for every data or MC processing
 - The analysts should be free to choose their own parameters for the corrections
 - The ntuples for the common performance modes should be available for everyone in a conventional format
 - Largely inspired by LHCb [PIDCalib framework](#).
- Our framework works with the following modes:
 - $D^{*+} \rightarrow [D^0 \rightarrow K^- \pi^+] \pi^+$ for K/ π ID
 - $\Lambda^0 \rightarrow p \pi^-$ for p/ π ID
 - $K_S \rightarrow \pi^+ \pi^-$ for π ID
 - $[\tau \rightarrow 3 \pi \nu] [\tau \rightarrow l \nu \nu]$ for lepton ID
- Fit mass distributions for each weighted performance mode and compute *sWeights* and signal-like histograms
 - Compute efficiencies, ROC curves, and data/MC weights, etc.



Ntuple production workflow for Weighted models



Framework mass fitting



- Mass fits for HadronID studies have been significantly improved
 - Reduced low-multiplicity background and improved signal shape
- The mass fit is used to produce sWeights for background subtraction
- In development:
 - $J/\psi \rightarrow \mu^+ \mu^- / e^+ e^-$ tag & probe, weighted study
 - $e^+ e^- \rightarrow \mu^+ \mu^- \gamma$ tag & probe, unweighted study

Ntuple production workflow for Weighted studies

- Create `settings.json` configuration file:

```
{
  "gbasf2_install_directory" : "~/gbasf2_install",
  "gbasf2_print_status_updates" : true,
  "gbasf2_max_retries" : 5,
  "gbasf2_cputime" : 5,
  "gbasf2_release" : "release-05-00-00",
  "gbasf2_download_logs" : false,
  "particleid_tasks" : "pid_task_list.yaml"
}
```

- Run Dataset Searcher and write down the LPNs to a `.list` file
- Create a workflow configuration file
- Launch command:
 - `python3 -m syscorr fw`
- The final weighted ntuples contain 2 sWeights columns:
 - A nominal column and a systematic one
 - Enables computation of systematic uncertainties

```
---
# ===== #
# Hadron ID task submission file #
# ===== #
-
  path: "my_lpns.list"
  proc: "bucket14"
  short_name: "Dst_b14"
  model: "Dst"
  cms_energy: "4S"
-
  path: "my_lpns.list"
  proc: "bucket14"
  short_name: "L0_b14"
  model: "Lambda0"
  cms_energy: "4S"
```

Fixed cut weights workflow


- This new workflow produces data/MC correction tables which are suitable for ParticleWeighting module and PIDVar
- Features:
 - Compact configuration file
 - Any number of efficiency cuts on global and/or binary Hadron ID
 - Tables can have any dimensionality and binning
 - Arbitrary preselection cuts
 - kaonID and protonID weights can be produced in parallel
 - [Documentation is online](#)
- Systematic uncertainties are available
- TODO:
 - Create a possibility to automatically upload the weights to CDB

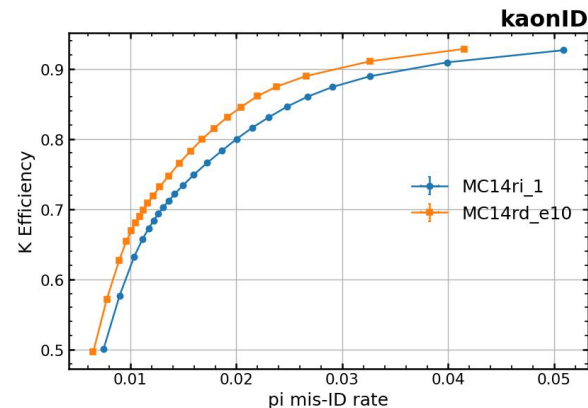
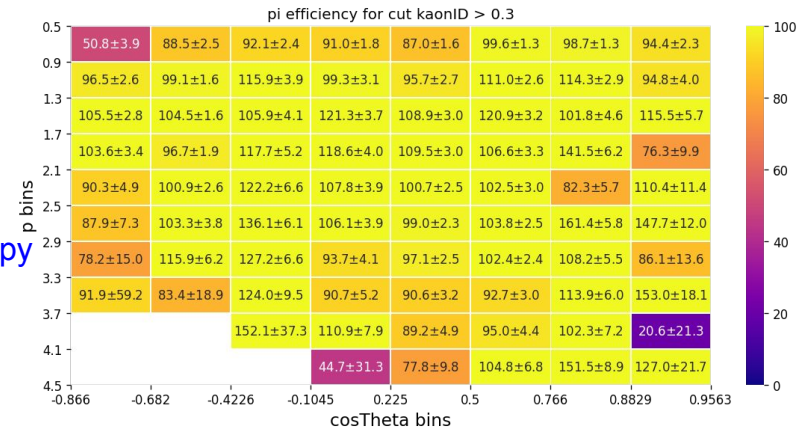


```
---  
# ===== #  
# Hadron ID weight configuration file #  
# ===== #  
weight_dir: 'fixed_weights/'  
remove tmp files: True  
weight_cfg_list:  
-  
  prefix_name: Rdtmc_v1  
  efficiency_particle_type: 'K'  
  fakerate_particle_type: 'pi'  
  binning: [[0.5, 2.5, 4.5],  
            [-0.8, 0.2, 0.9563]]  
  track_variables: [ "p", "cosTheta" ]  
  cuts: [ "kaonID > 0.2", "kaonID > 0.8" ]  
  precuts: [ "", "charge > 0", "charge < 0" ]  
  mc_proc_query: [ "MC14ri_1" ]  
  data_proc_query: [ "procl2e7" ]
```

- Launch command:
 - `python3 -m syscorr fw -w fixed_weights`

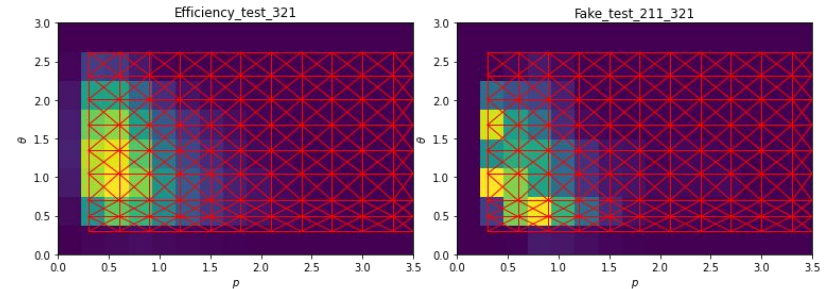
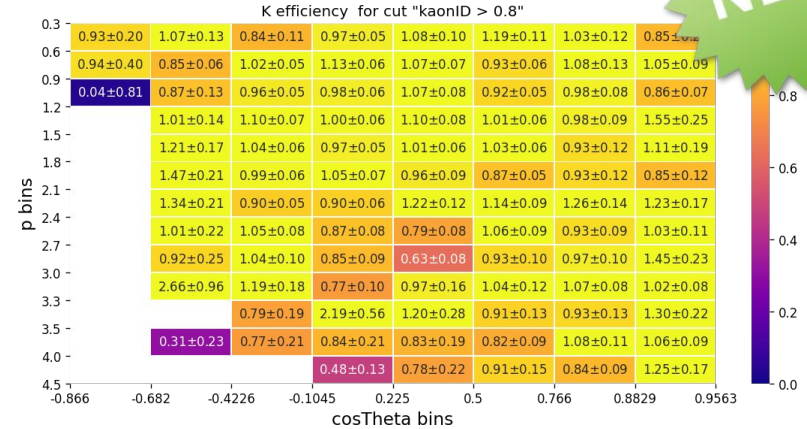
User scripts

- The script folder at KEKCC:
 - systematic_corrections_framework/scripts/
- Physics analysis scripts:
 - NEW** Data/MC weights `weight_table.py`
 - Multitrack calibration `perform_multitrack_calibration.py`
 - NEW** KDE PID fit `perform_pid_fit.py`
- Performance scripts:
 - Efficiency table `efficiency_table.py`
 - ROC curve `id_vs_misid_curve.py`
 - NEW** Histogram `plot_distribution.py`
 - Result combiner script `process_tables.py`
- Support scripts:
 - Show ntuple DB content `show_db_content.py`
 - NEW** Print variables `show_variables.py`
- There is no need to install the framework to run them
- User scripts can be launched in the following ways:
 - Command line  Useful for workflow management, e.g. snakemake, b2luigi
 - **In the IPython mode**



Interaction with PIDVar

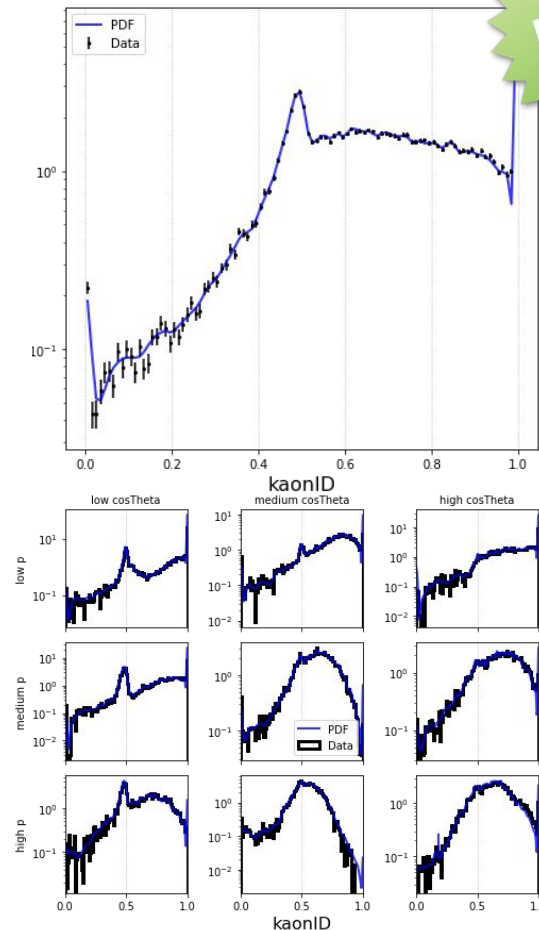
- One can produce the weights in a notebook and immediately apply them to MC ntuple
 - The data/MC ratios converted into weight table format via `create_weights()`
- The application of the weights is done using PIDVar framework
 - Requires efficiency and fake rate weights
 - Provides uncertainties on MC ntuples
- **The weight tables from the framework need to be adjusted for PIDVar**
 - E.g. renaming columns, converting column values, etc.
- [Tutorial is online](#)
- It is strongly recommended to save the weight tables for the analysis reproducibility



Multidimensional PID Fit

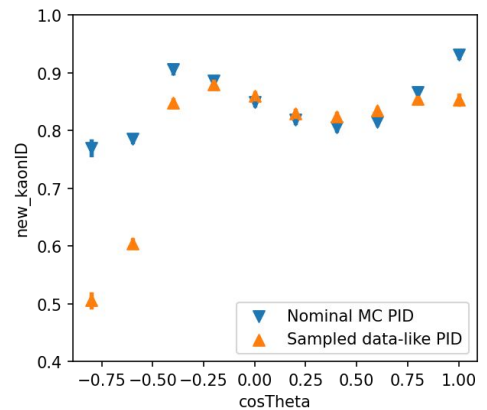
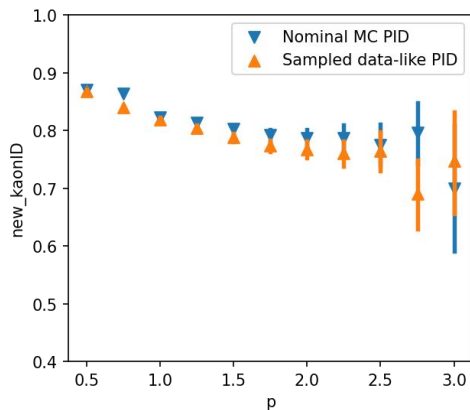
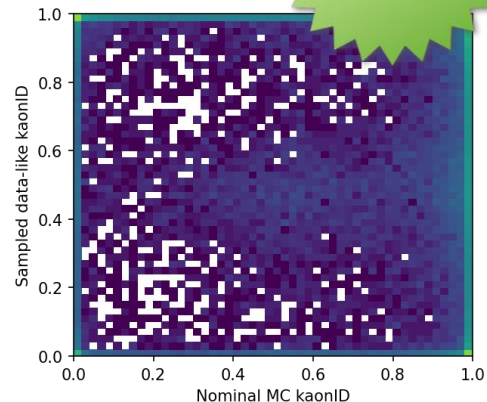
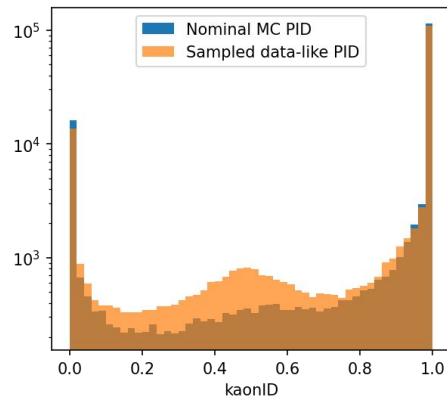
NEW

- An alternative to the reweighting of the histograms
- One can fit real data PID as function of $(p, \cos\theta, \dots)$ and sample from it using MC $(p, \cos\theta, \dots)$
 - Sampled variable on MC would be data-like PID
 - Implemented in LHCb
 - Free choice of cut values by the analysts
 - Much more granularity than in the weight tables
 - e.g. 200 bins for PID, 100 bins for $(p, \cos\theta, \dots)$
 - Sampled variable can be used in ML algorithms
- Implemented using LHCb [KDE Meerkat](#) library
- Free choice of the fit dimensionality and the variables
- The output of the script is saved as ROOT file
- [Tutorial is online](#)



PID resampling

- The ROOT files from the PID KDE fit are used to sample the data-like PID on MC
- The resampled PID on MC ntuples will have data-like efficiency and fake rates as well as realistic profiles
- First validation studies have been performed
- [Tutorial is online](#)
- Coming soon™:
 - Systematic uncertainties for the method
 - Instead of resampling one can transform MC PID and preserve the 1st order of correlations
 - Upload output ROOT files to CDB
 - Create basf2 module that can sample or transform MC PID during runtime



Cheatsheet

- Documentation: [link](#)
- Confluence: [link](#)
- Git repository: [link](#)
- Bamboo: [link](#)

- Ntuples on KEKCC:

- `/group/belle2/dataproduct/Systematics/production/`

- Ntuple structure:

- Follows the `vu.create_aliases_for_selected()` pattern

- Available datasets can be shown by `scripts/show_db_content.py`:

- `proc12` + prompt up to `bucket25`, `sproc2`, `3`
- 2 blocks of MC14ri
- MC14rd exp7-10 + buckets (being updated)



- Available variables can be shown by `scripts/show_variables.py`:

- Now includes Charged BDT and track isolation variables

SysCorrFW 0.4.3
documentation

Q Search the docs ...

CONTENTS:

User scripts
PID studies
Installation
Tutorials
Available `b2lujr` task classes
Available `basf2` modules
Other ParticleID modules
Changelog



Systematic Corrections Framework

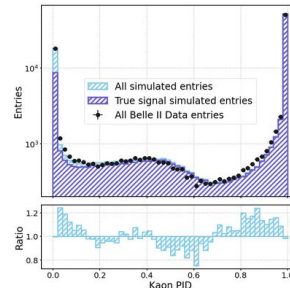
Differences between simulation and experimental data are one of the main components of a typical systematic uncertainty of any physics analysis. For example, selection of charged particles by particle identification (PID) will result in different selection efficiencies on the simulation and experimental data samples in most of the cases, the differences are demonstrated in the figure below. To reduce the systematic uncertainty, one can transform the simulated PID distributions or compute the efficiencies directly on data, which is the main purpose of this framework.

Contents:

- User scripts
- PID studies
- Installation
- Tutorials
- Available `b2lujr` task classes
- Available `basf2` modules
- Other ParticleID modules
- Changelog

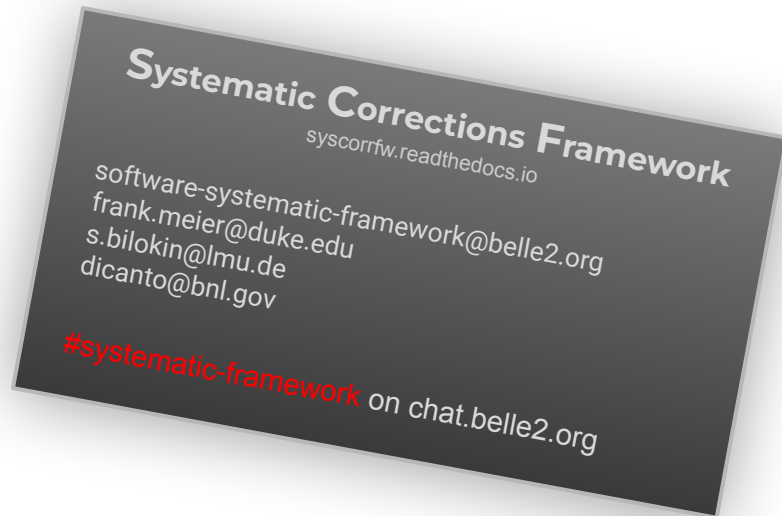
Basics

Most of physics analyses in Belle II rely on



Summary

- This framework is designed to automate the performance studies and computation of systematic weights associated to PID
- Progress so far:
 - ✓ Added a possibility to integrate other types of studies
 - ✓ Processed proc12 + all buckets up to 25th + s-proc's
 - ✓ Introduced PID KDE Fit in LHCb fashion
 - Integration of Lepton ID modes is in progress
 - ☒ Create meta-variables in basf2 fashion
 - ☒ Duplicate dataset to other servers, e.g. BNL or DESY
 - ☒ Integration with B2Production framework
 - ☒ Integration with basf2 and b2conditiondb

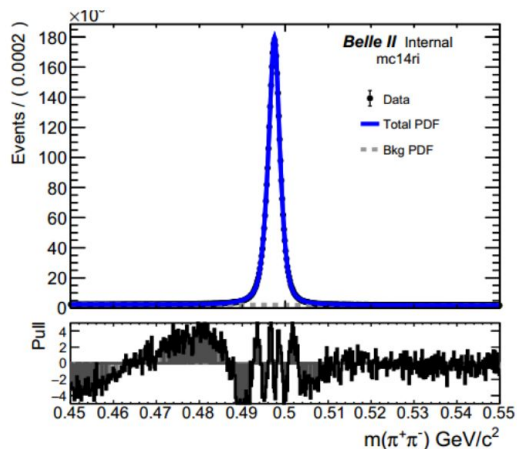


- Please contact us if you want to contribute:
 - Framework development
 - Producing ntuples and testing new features
 - Integrating your performance studies
 - Validation of weights and procedures

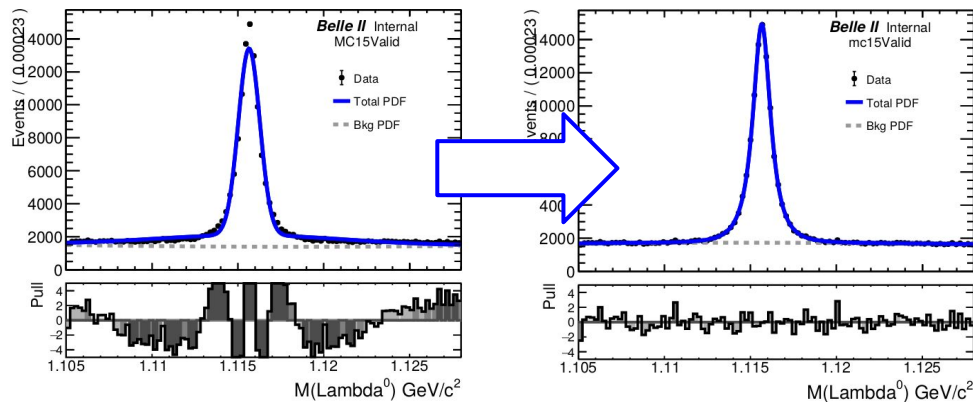
Thank you!

Model overview

- The $K_S \rightarrow \pi^+\pi^-$ has been integrated into the framework:
 - [BELLE2-NOTE-TE-2019-024]
- PDF:
 - Signal: RooJohnson + RooGaussian
 - Bkg: 3rd order Chebychev
- Work to improve the PDF model is ongoing

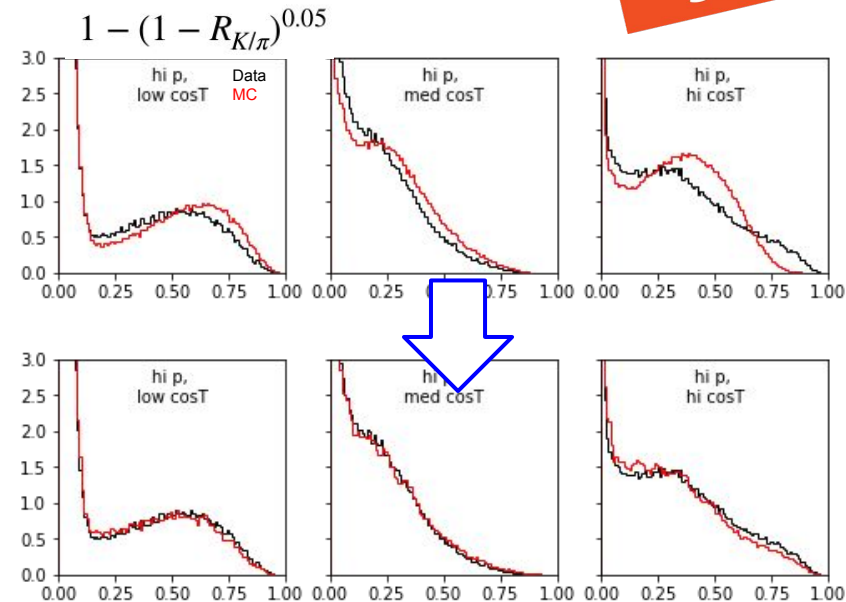
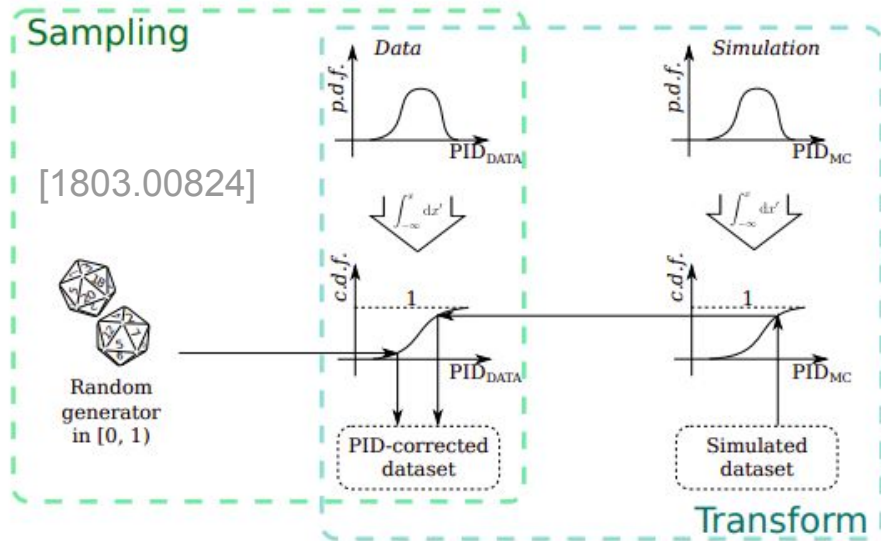


- The $\Lambda^0 \rightarrow p\pi^-$ has been overhauled
- PDF:
 - Signal: RooJohnson
 - Bkg: 2nd order Chebychev
- Ntuples have been reprocessed already



User scripts: [BIIPERF-143]

COMING
SOON!



- Transforming / resampling of the MC PID distributions [Anton's talk last B2GM]
- Usage of an old external C++ library for KDE fitting slows down the development
 - Looking for alternatives in python: N-dimensional KDE fit with weight and custom kernel support, e.g. KDEpy
- It should be possible to integrate the algorithm into basf2 and b2conditionsdb, help needed

Refit workflow for HadronID

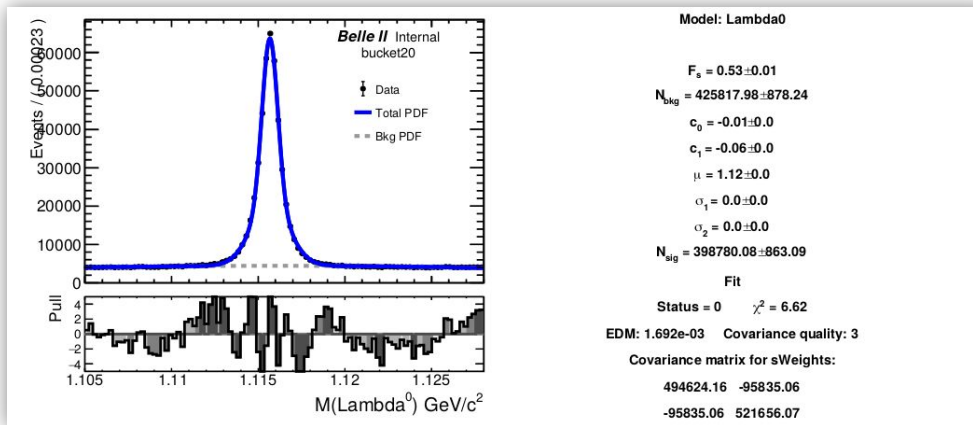
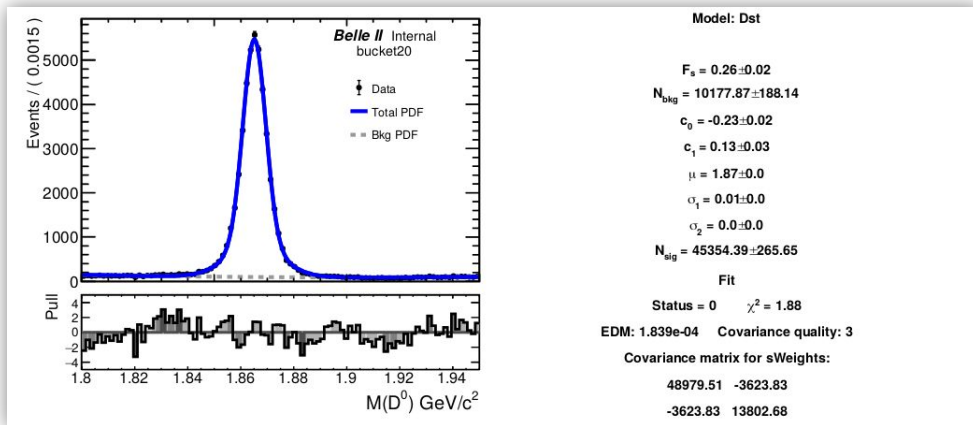


- This new workflow reproduces sWeights in already existing ntuples in case if a PDF model has been updated
- Features:
 - Compact configuration file
 - One can select a particular model to update
 - The ntuple files will acquire a new suffix and their sWeights column will be replaced
 - Workflow will also reproduce all benchmark plots
- Will be applied when we will update Lambda0 fit

- Launch command:
 - `python3 -m syscorr fw -w refit`

```
---  
# ===== #  
# Hadron ID refit configuration file #  
# ===== #  
old_suffix: ""  
new_suffix: "_v1"  
models: ["Lambda0", "Dst"]
```


Ntuple production workflow for HadronID



- Path on KEKCC:
 /group/belle2/dataproduct/Systematics/
- Ntuples are saved in the
 production/{proc}/ directories
 - `vu.create_aliases_for_selected()`
 pattern
- After every run the plots are saved to
 the production/plots/ directory
- **Fitted function and the fit results are
 saved to production/fit_results/
 directory**
- **Validation or benchmark plots are saved
 as json to production/validation/**



How to add a new hadron ID study

1. Create skimming for the physics mode
2. Create PID model class:

```
class MyExamplePIDModel(PIDModelBase):  
    def __init__(self):  
        self.m_version = 0.01  
        self.m_mc_truth_variable = 'B0_isSignal'
```

- Add information for the ntuple production
- Add PDF model for fitting

```
def get_ntuple_model(self) -> dict:  
    result = {'list_name': 'X:p_list_name',  
             'selection_cuts': 'E > 0'}  
    result['variables'] = ['M', 'pionID']  
    result['hist'] = [('M', 100, 1.75, 2.0)]  
    result['hist_2d'] = []  
    return result
```

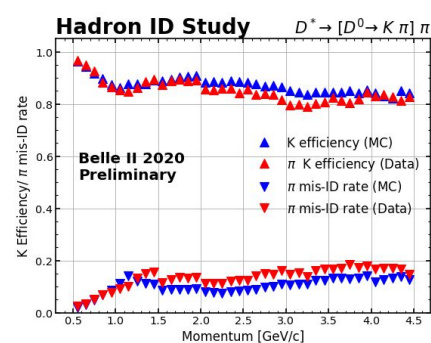
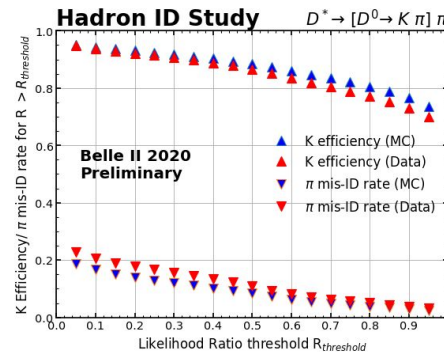
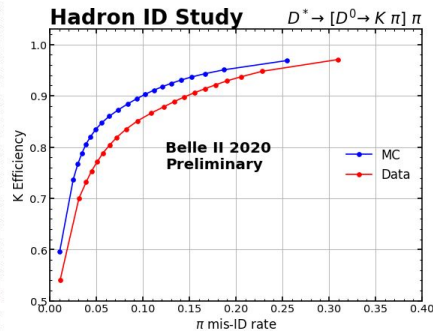
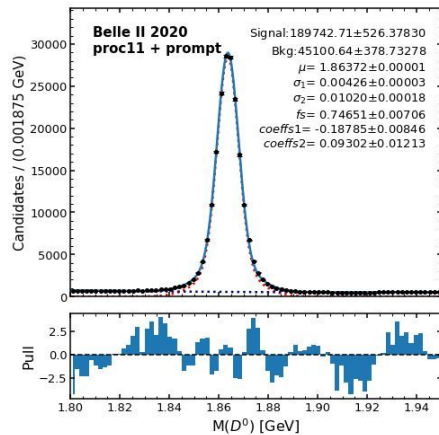
- Returns dict of skimmed list name, selection cuts and list of variables
- One can use variable manager to create aliases

```
def get_roofit_pdf_model(self, yields: tuple, var_stash: list) -> tuple:  
    mass = ROOT.RooRealVar('M', 'm_{inv} GeV/c', 1.75, 2.)  
    # Signal PDF:  
    mean = ROOT.RooRealVar('mean', 'mean', 1.8, 1.75, 2.)  
    sigma = ROOT.RooRealVar('sigma', 'sigma', 0.01, 0.001, 1.)  
    signal = ROOT.RooGaussian('signal', 'signal', mass, mean, sigma)  
    # Background PDF:  
    par = ROOT.RooRealVar('bkg_par', 'bkg_par', -3, -10, 0)  
    bkg = ROOT.RooExponential('bkg', 'bkg', mass, par)  
    # Save variables:  
    addToStash(var_stash, [mean, sigma, signal, par, bkg])  
    # Combined model:  
    model = ROOT.RooAddPdf('model', 'model', ROOT.RooArgList(signal, bkg),  
                           ROOT.RooArgList(*yields))  
    return model, mass
```

- Has to return tuple of PDF and mass variable

D* K/ π ID study

- D* reconstruction note:
 - S. Sandilya BELLE2-NOTE-PH-2019-048
- Default cuts in the framework
 - impact parameters $dr < 2$, $abs(dz) < 4$ and CDC hits > 20 ;
 - $p_{D^*}^{cms} > 2.5$ GeV;
 - 0.1439 GeV $< \Delta M < 0.1469$ GeV;
 - 1.8 GeV $< M(D^0) < 1.95$ GeV.
- Model for $M(D^0)$
 - Signal: two Gaussian functions with a common mean;
 - Background: the second order Chebychev function.
- The full set of global PID variables (pionID, kaonID, etc) and the likelihoods for each mass hypothesis and detector is stored.



Λ^0 p/π ID study

- Λ reconstruction note
 - B. Scavino BN-2020-027
- Two different skim types
 - Analysis skim: $0.6 < p_p / p_\Lambda < 1.0$, $\text{flightSignificance} > 3.0$, $\text{protonID} > 0.1$, $\cos(\alpha) > 0.99$
 - HLT skim: $\text{flightSignificance} > 10.0$, $(p_p - p_\pi)/(p_p + p_\pi) > 0.41$
 - Both skims slightly different than our selection, analysis skim is similar to Todd's studies: $n\text{CDCHits} > 20$, $p_p / p_\Lambda > 0.6$, $\text{flightSignificance} > 2$
- ✓ Performance of sWeights relative to MC truth matching has been validated

