Karlsruhe Institute of Technology

Institute of Experimental Particle Physics

b2luigi - Bringing Batch 2 luigi!

PyHEP.dev 2024 - "Python in HEP" Developer Workshop

Alexander Heidelbach, Jonas Eppelt
alexander.heidelbach@kit.edu

luigi in a Nutshell

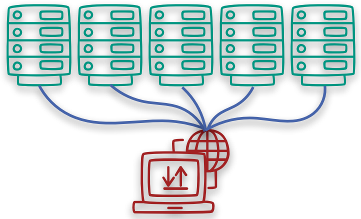
- Building a Pipeline
  - Dependency Resolution
  - Workflow Management
  - Visualisation
  - Handling Failures
  - Command Line Integration
  - …
- https://github.com/spotify/luigi

"Hello World" in luigi:

```
class MyTask(luigi.Task):
        parameter = luigi.Parameter()

        def run(self):
                do_smth(self.parameter)

        def output(self):
                return Target("some/file")

        def requires(self):
                yield OtherTask()
```

Why b2luigi?

- Many many many jobs!
  - Running batch job = running process
  - Problem: Limitation on the number of processes per user
  - Solution: Single process on submission machine
- Many many many tasks!
  - Problem: Tasks in luigi need to adjust for batch execution specifically
  - Solution: Abstract batch submission away from the task
- Which batch system?
  - Problem: Batch system usage defined by task instance
  - Solution: Batch system usage only defined by config variable



`b2luigi.process(…,batch=True)`

Before we dive into it…

- b2luigi was written by a group of PhD students for their analyses
  - Goal: Make everyday work a little bit easier
  - Not a goal: Invent the next workflow management system
    - A lot of helper functions on top of luigi
    - Easy transition between luigi and b2luigi

- Since this year Belle II has been the official maintainer
  Completely new team of developers
  Targeting collaboration (and beyond…) wide use

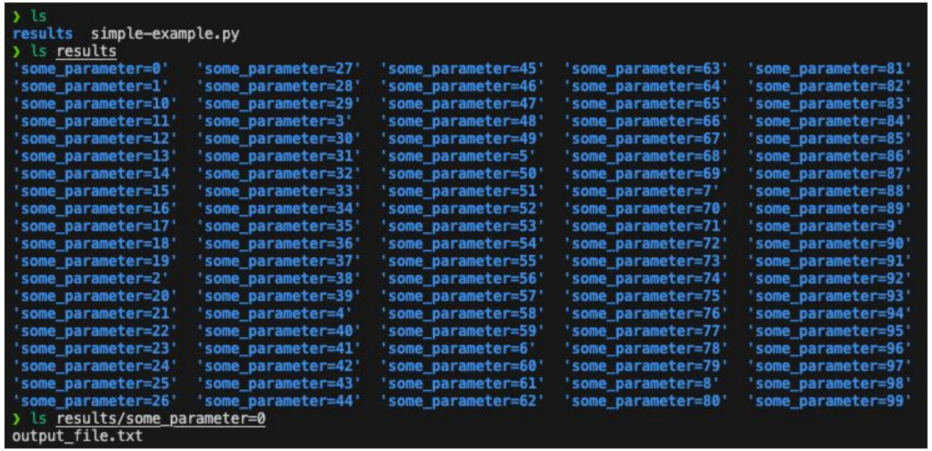
b2luigi vs luigi

```
import luigi

class MyTask(luigi.Task):
    parameter = luigi.Parameter()

    def run(self):
        with open(f“/my/target/dir/my_output{self.parameter}.file“, "w") as f:
            f.write(f“{self.parameter}“)

    def output(self):
        return luigi.Target(f“/my/target/dir/my_output{self.parameter}.file“)

    def requires(self):
        yield OtherTask()

if __name__ == “__main__“:
    luigi.build(
        [MyTask(some_parameter=i) for i in range(100)],
        workers=20
    )
```

```
import b2luigi as luigi

class MyTask(luigi.Task):
    parameter = luigi.Parameter()

    def run(self):
        with open(self.get_output_file_name("my_output.file"), "w") as f:
            f.write(f“{self.parameter}“)

    def output(self):
        yield self.add_to_output(“my_output.file”)

    def requires(self):
        yield OtherTask()

if __name__ == “__main__“:
    luigi.set_setting(“result_dir”, “/my/target/dir/”)
    luigi.process(
        [MyTask(some_parameter=i) for i in range(100)],
        workers=20
    )
```

Automated Output Bookkeeping

```
===== Luigi Execution Summary =====

Scheduled 100 tasks of which:
* 100 ran successfully:
   - 100 MyTask(some_parameter=0,1,10,11,12,13,14,15,16,17,18,...)

This progress looks :) because there were no failed tasks or missing dependencies

===== Luigi Execution Summary =====
```



```
❯ ls
results  simple-example.py
❯ ls results
'some_parameter=0'   'some_parameter=27'  'some_parameter=45'  'some_parameter=63'  'some_parameter=81'
'some_parameter=1'   'some_parameter=28'  'some_parameter=46'  'some_parameter=64'  'some_parameter=82'
'some_parameter=10'  'some_parameter=29'  'some_parameter=47'  'some_parameter=65'  'some_parameter=83'
'some_parameter=11'  'some_parameter=3'   'some_parameter=48'  'some_parameter=66'  'some_parameter=84'
'some_parameter=12'  'some_parameter=30'  'some_parameter=49'  'some_parameter=67'  'some_parameter=85'
'some_parameter=13'  'some_parameter=31'  'some_parameter=5'   'some_parameter=68'  'some_parameter=86'
'some_parameter=14'  'some_parameter=32'  'some_parameter=50'  'some_parameter=69'  'some_parameter=87'
'some_parameter=15'  'some_parameter=33'  'some_parameter=51'  'some_parameter=7'   'some_parameter=88'
'some_parameter=16'  'some_parameter=34'  'some_parameter=52'  'some_parameter=70'  'some_parameter=89'
'some_parameter=17'  'some_parameter=35'  'some_parameter=53'  'some_parameter=71'  'some_parameter=9'
'some_parameter=18'  'some_parameter=36'  'some_parameter=54'  'some_parameter=72'  'some_parameter=90'
'some_parameter=19'  'some_parameter=37'  'some_parameter=55'  'some_parameter=73'  'some_parameter=91'
'some_parameter=2'   'some_parameter=38'  'some_parameter=56'  'some_parameter=74'  'some_parameter=92'
'some_parameter=20'  'some_parameter=39'  'some_parameter=57'  'some_parameter=75'  'some_parameter=93'
'some_parameter=21'  'some_parameter=4'   'some_parameter=58'  'some_parameter=76'  'some_parameter=94'
'some_parameter=22'  'some_parameter=40'  'some_parameter=59'  'some_parameter=77'  'some_parameter=95'
'some_parameter=23'  'some_parameter=41'  'some_parameter=6'   'some_parameter=78'  'some_parameter=96'
'some_parameter=24'  'some_parameter=42'  'some_parameter=60'  'some_parameter=79'  'some_parameter=97'
'some_parameter=25'  'some_parameter=43'  'some_parameter=61'  'some_parameter=8'   'some_parameter=98'
'some_parameter=26'  'some_parameter=44'  'some_parameter=62'  'some_parameter=80'  'some_parameter=99'
❯ ls results/some_parameter=0
output_file.txt
```

- The self.add_to_output function automatically ensures each output file is unique
- The task's parameter are used to construct a path structure:
  - result_dir
    - some_parameter=1
      - my_output.file
    - some_parameter=2
      - my_output.file
    - ….
- Use luigi.Parameter's significant and hash_function, as well as the declaration order to control this behaviour

Batch Processing

- Initial motivation: Make batch submission easy!
- Currently fully supported:
  - HTCondor
  - LSF
  - Gbasf2 (BelleII@WLCG)
- Challenges:
  - Using your environment
    - Settings: env_script, env,..
  - Ensuring consistent locations
    - Settings: working_dir, result_dir,…

```
class MyTask(b2luigi.Task):
    batch_system = "htcondor"
```

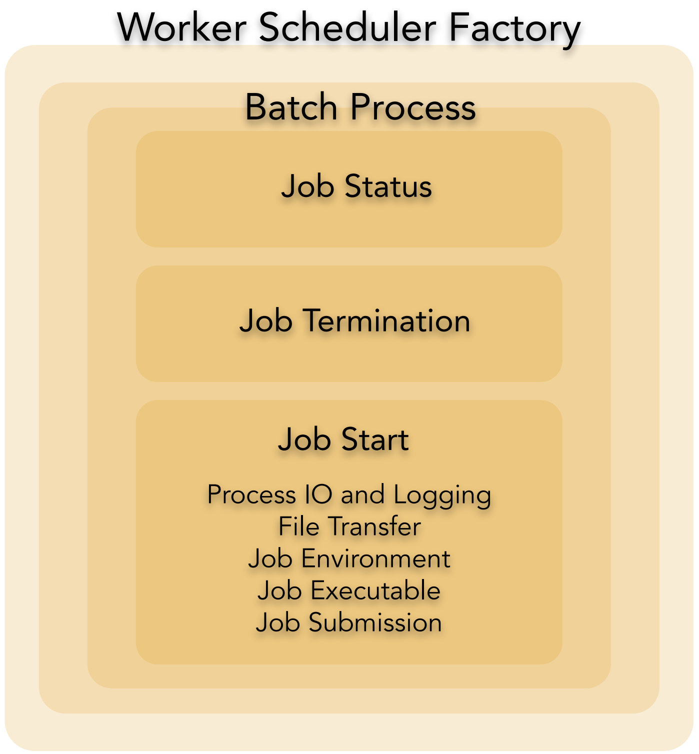
```
class MyTask(b2luigi.Task):
    @property
    def htccondor_settings(self):
        return {"request_memory": 4096}
```

```
b2luigi.set_setting("result_dir", "path/to/result")
```

```
settings.json
{
    "env_script": "setup.sh"
}
```

Batch Processing in b2luigi

- Use of luigi's Worker Scheduler Factory
- Different processes for each batch system
- Batch process handles:
  - Job status
  - Job start
  - Job termination
- Job start consists (mostly) of two steps:
  - Creation of executable wrapper
  - Submission via batch system mechanism



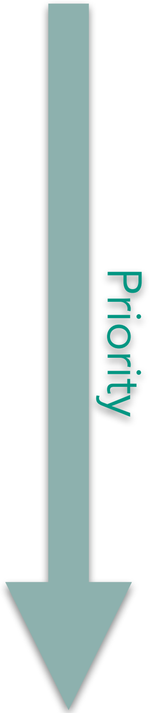
Task Settings

```
class MyTask(b2luigi.Task):
    some_setting = “some value”
```

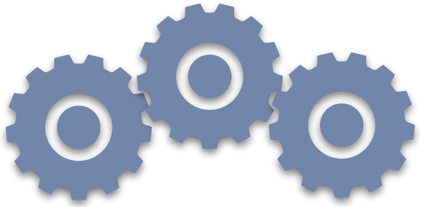
```
class MyTask(b2luigi.Task):
    @property
    def some_setting(self):
        return “some value”
```

```
b2luigi.set_setting(“some_setting”, "some value")
```

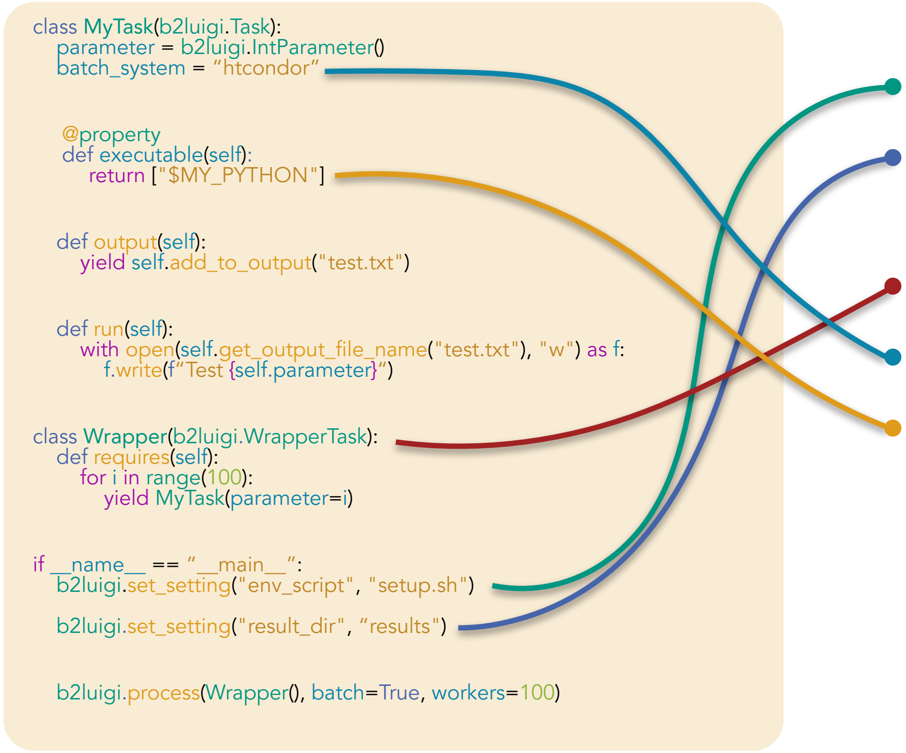
```
settings.json
{
    “some_setting”: “some value”
}
```



- Settings are handled by b2luigi!
- Control:
  - Batch system choice
  - Output and log path
  - Environment
  - Workflow specific settings



Example: HTCondor

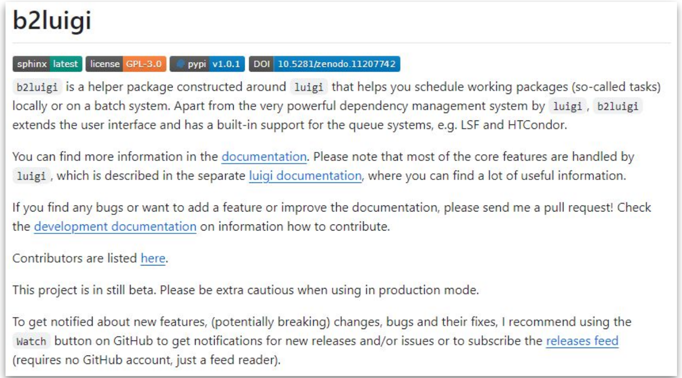


```
class MyTask(b2luigi.Task):
    parameter = b2luigi.IntParameter()
    batch_system = "htcondor"


    @property
    def executable(self):
        return ["$MY_PYTHON"]


    def output(self):
        yield self.add_to_output("test.txt")


    def run(self):
        with open(self.get_output_file_name("test.txt"), "w") as f:
            f.write(f"Test {self.parameter}")


class Wrapper(b2luigi.WrapperTask):
    def requires(self):
        for i in range(100):
            yield MyTask(parameter=i)


if __name__ == "__main__":
    b2luigi.set_setting("env_script", "setup.sh")

    b2luigi.set_setting("result_dir", "results")


    b2luigi.process(Wrapper(), batch=True, workers=100)
```

- Script executed on the batch job side
- Location needs to be accessible on the batch job side
- Wrapper tasks need no output
- Definition of the batch system
- Executable for this specific task
    - E.g. environment variable set in setup.sh

Summary & Outlook

- b2luigi provides a simple and flexible implementation to run your workflow on batch systems!
- The abstraction of the batch processing to global settings allows for:
  - Quick change in the submission strategy
  - Simple code and execution
- Outlook
  - Code Maintainability
  - XRootDTarget
  - Grid Tool API



b2luigi

sphinx latest | license GPL-3.0 | pypi v1.0.1 | DOI 10.5281/zenodo.11207742

`b2luigi` is a helper package constructed around `luigi` that helps you schedule working packages (so-called tasks) locally or on a batch system. Apart from the very powerful dependency management system by `luigi`, `b2luigi` extends the user interface and has a built-in support for the queue systems, e.g. LSF and HTCondor.

You can find more information in the documentation. Please note that most of the core features are handled by `luigi`, which is described in the separate luigi documentation, where you can find a lot of useful information.

If you find any bugs or want to add a feature or improve the documentation, please send me a pull request! Check the development documentation on information how to contribute.

Contributors are listed here.

This project is in still beta. Please be extra cautious when using in production mode.

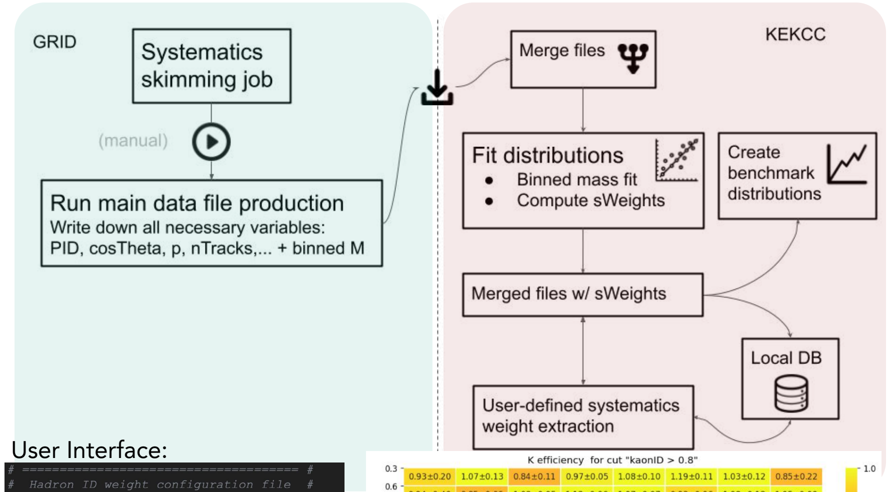
To get notified about new features, (potentially breaking) changes, bugs and their fixes, I recommend using the `Watch` button on GitHub to get notifications for new releases and/or issues or to subscribe the releases feed (requires no GitHub account, just a feed reader).

Github zenodo Zenodo

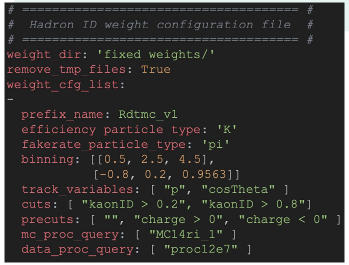
Documentation PyPi

Systematic Corrections Framework

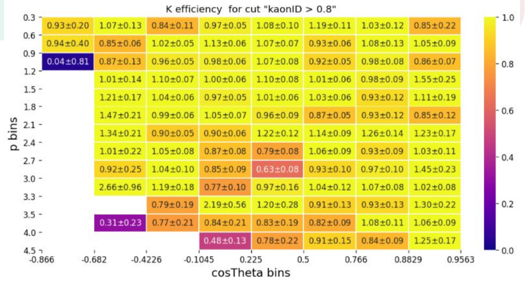
- 2 Stage Algorithm
  - Ntuple Production
    - Centrally run for every campaign
    - Running: Gbasf2
  - Data/MC Corrections
    - User runs their specific selection
    - Running: Locally, HTCondor, LSF
- Also: Validation of different datasets


GRID
Systematics skimming job
(manual)
Run main data file production
Write down all necessary variables:
PID, cosTheta, p, nTracks,... + binned M
KEKCC
Merge files
Fit distributions
Binned mass fit
Compute sWeights
Create benchmark distributions
Merged files w/ sWeights
Local DB
User-defined systematics weight extraction

User Interface:



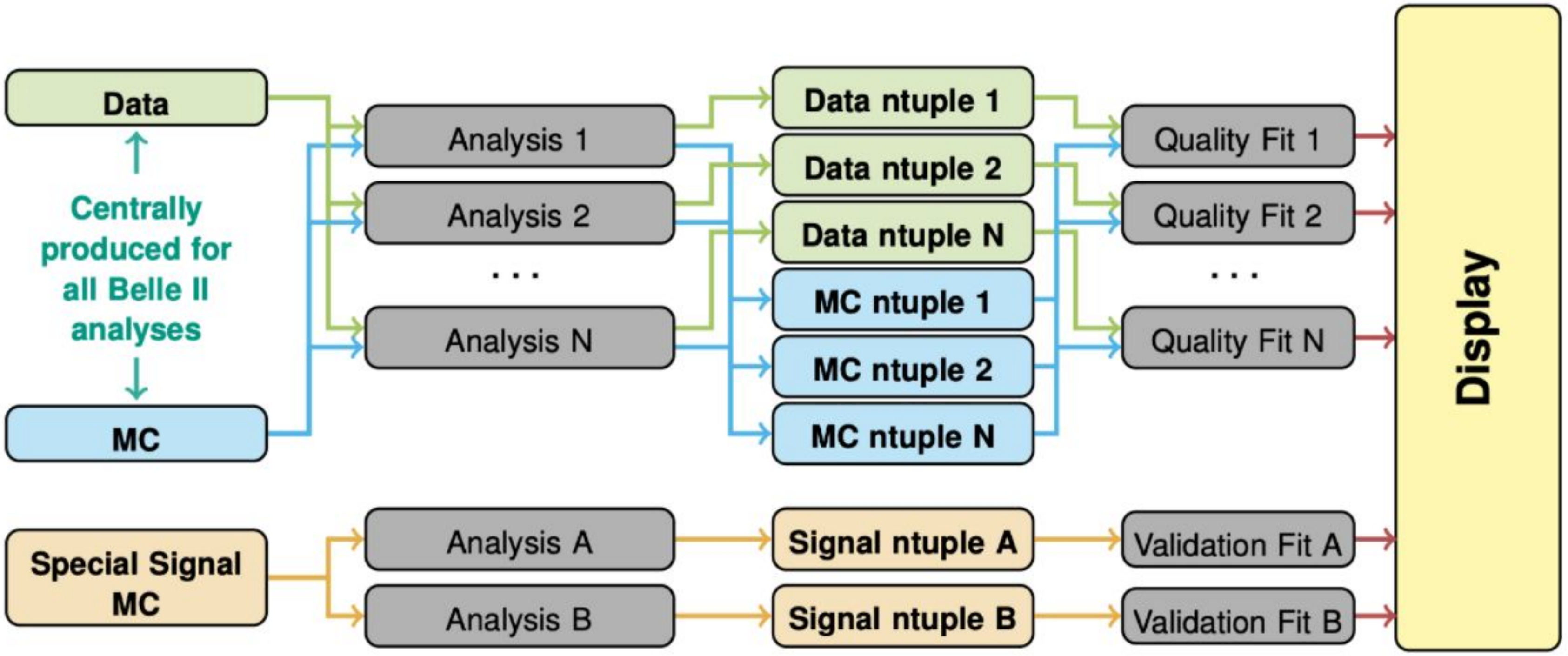
```
# ==================================== #
#  Hadron ID weight configuration file  #
# ==================================== #
weight_dir: 'fixed_weights/'
remove_tmp_files: True
weight_cfg_list:
-
  prefix_name: Rdtmc_v1
  efficiency particle type: 'K'
  fakerate_particle_type: 'pi'
  binning: [[0.5, 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: [ "proc12e7" ]
```



p bins \ cosTheta bins	-0.866 – -0.682	-0.682 – -0.4226	-0.4226 – -0.1045	-0.1045 – 0.225	0.225 – 0.5	0.5 – 0.766	0.766 – 0.8829	0.8829 – 0.9563
0.3 – 0.6	0.93±0.20	1.07±0.13	0.84±0.11	0.97±0.05	1.08±0.10	1.19±0.11	1.03±0.12	0.85±0.22
0.6 – 0.9	0.94±0.40	0.85±0.06	1.02±0.05	1.13±0.06	1.07±0.07	0.93±0.06	1.08±0.13	1.05±0.09
0.9 – 1.2	0.04±0.81	0.87±0.13	0.96±0.05	0.98±0.06	1.07±0.08	0.92±0.05	0.98±0.08	0.86±0.07
1.2 – 1.5		1.01±0.14	1.10±0.07	1.00±0.06	1.10±0.08	1.01±0.06	0.98±0.09	1.55±0.25
1.5 – 1.8		1.21±0.17	1.04±0.06	0.97±0.05	1.01±0.06	1.03±0.06	0.93±0.12	1.11±0.19
1.8 – 2.1		1.47±0.21	0.99±0.06	1.05±0.07	0.96±0.09	0.87±0.05	0.93±0.12	0.85±0.12
2.1 – 2.4		1.34±0.21	0.90±0.05	0.90±0.06	1.22±0.12	1.14±0.09	1.26±0.14	1.23±0.17
2.4 – 2.7		1.01±0.22	1.05±0.08	0.87±0.08	0.79±0.08	1.06±0.09	0.93±0.09	1.03±0.11
2.7 – 3.0		0.92±0.25	1.04±0.10	0.85±0.09	0.63±0.08	0.93±0.10	0.97±0.10	1.45±0.23
3.0 – 3.3		2.66±0.96	1.19±0.18	0.77±0.10	0.97±0.16	1.04±0.12	1.07±0.08	1.02±0.08
3.3 – 3.5			0.79±0.19	2.19±0.56	1.20±0.28	0.91±0.13	0.93±0.13	1.30±0.22
3.5 – 4.0		0.31±0.23	0.77±0.21	0.84±0.21	0.83±0.19	0.82±0.09	1.08±0.11	1.06±0.09
4.0 – 4.5				0.48±0.13	0.78±0.22	0.91±0.15	0.84±0.09	1.25±0.17

Documentation

Validation Interface for the Belle II Experiment


Data
Centrally produced for all Belle II analyses
MC
Analysis 1
Analysis 2
. . .
Analysis N
Data ntuple 1
Data ntuple 2
Data ntuple N
MC ntuple 1
MC ntuple 2
MC ntuple N
Quality Fit 1
Quality Fit 2
. . .
Quality Fit N
Special Signal MC
Analysis A
Analysis B
Signal ntuple A
Signal ntuple B
Validation Fit A
Validation Fit B
Display