HEPData and Snakemake

Slavomira Stefkova

Lectures/Flavour Physics IPHC 2023
10.11.2023

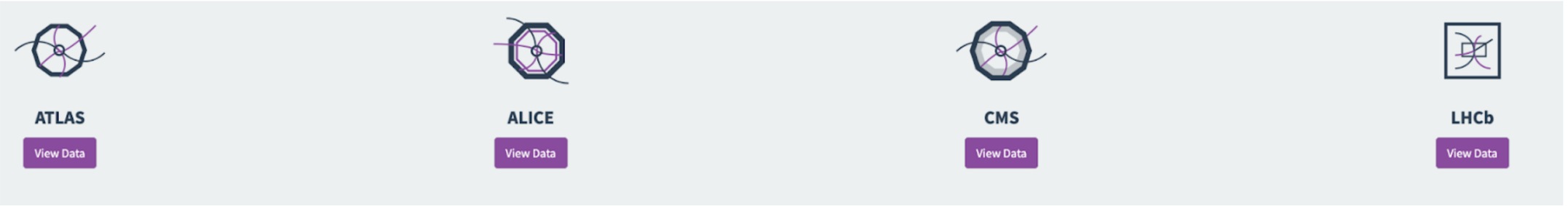
HEPData: What? Why? How?

What is HEPData?

https://www.hepdata.net/

What is HEPData?

- Open-access repository for high-energy physics data (40 years old)
- Funded by STFC (UK) and is based at the IPPP at Durham university
- All big LHC experiments publish their data and use this platform rather frequently, for many its part of their publication checklist.



What is HEPData?

What are the main purposes of HEPData?

1. To facilitate data preservation

1

What is HEPData?

What are the main purposes of HEPData?

1. To facilitate data preservation
2. To give everyone access to HEP results in data format

1

2

What is HEPData?

What are the main purposes of HEPData?

1. To facilitate data preservation
2. To give everyone access to HEP results in data format
3. To serve as a platform for physics reinterpretation

1

2

3

What is HEPData?

What are the main purposes of HEPData?

1. To facilitate data preservation
2. To give everyone access to HEP results in data format
3. To serve as a platform for physics reinterpretation

1

2

3

HEPData = digitised plots from the publications + anything that you deem important so that it serves these purposes

High Energy Physics and HEPData

LHCb and Belle II are part of HEPData:

- LHCb coordinator mailing list: alex.grecu@cern.ch
- Belle II coordinator mailing list: mgt-hepdata-coordinators@belle2.org

HEPData

Repository for publication-related High-Energy Physics data

What both of the collaborations should do wrt to HEPData:

- internal guidance on **how/what** to publish on HEPData
- make HEPData Record **part of the publication checklist**

[Talk by Graeme Watt]

Fraction of HEPData publications/total publication

ALICE: **92%**
ATLAS: **57%**
CMS: **49%**
LHCb: **13%**

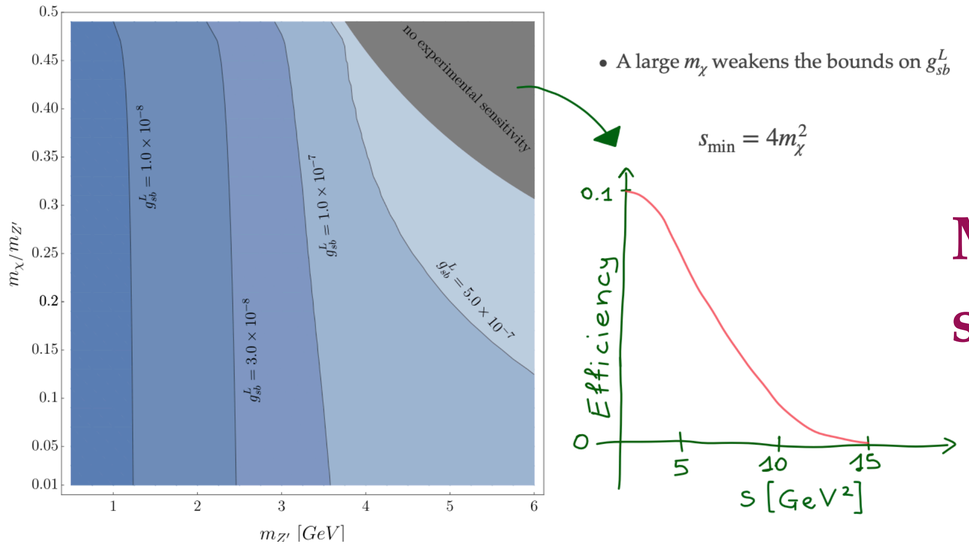
Belle: 31/533 = **6%**
Belle-II: 1/9 = **11%**

Why is this so important?

A real life example from C. Manzari Flavour @ Crossroads Mainz 2022 talk

Light Z'

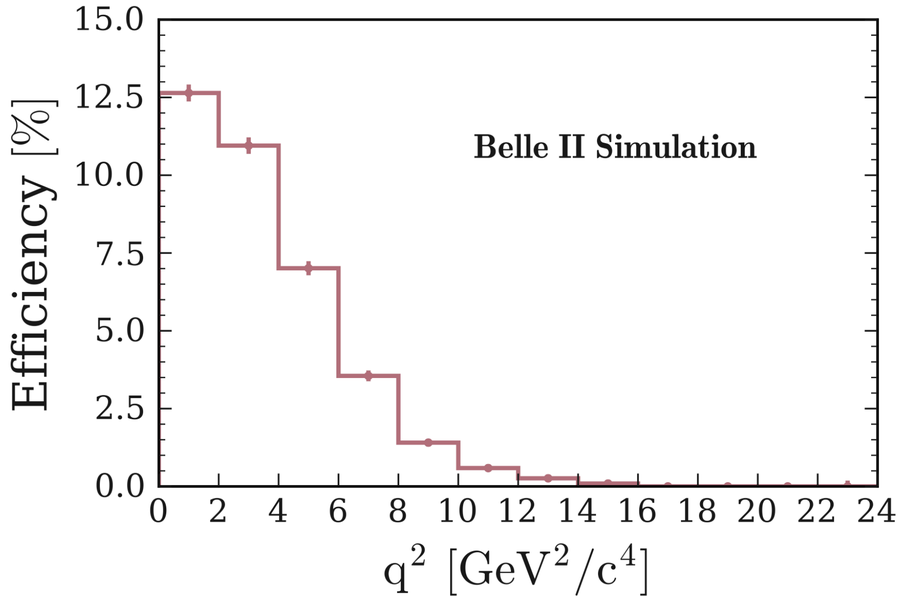
Recasted Belle II analysis



HEPData

Minimise errors + speed up process

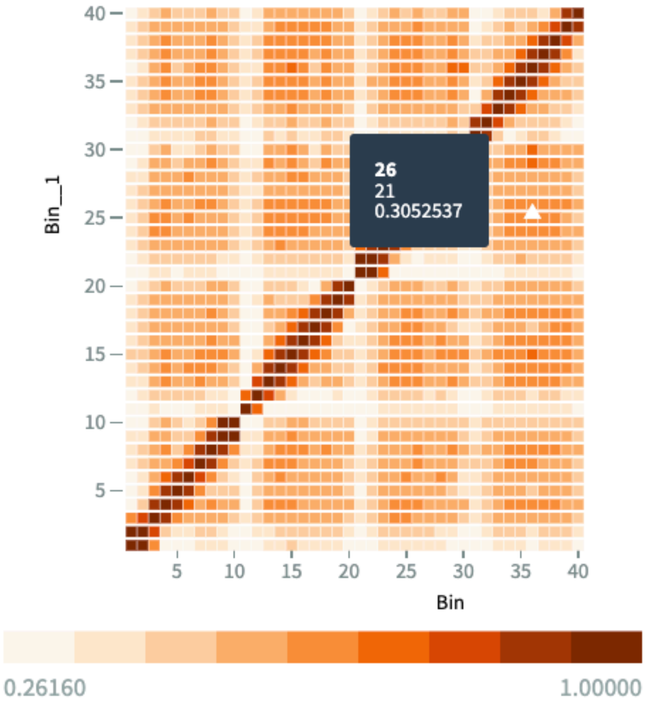
Figure 3 from PRL 127, 181802 (2021)



Why is this so important?

Belle Example

Precise determination of the CKM matrix element $|V_{cb}|$ with $\bar{B}^0 \to D^{*+} \ell^- \bar{\nu}_\ell$ decays with hadronic tagging at Belle



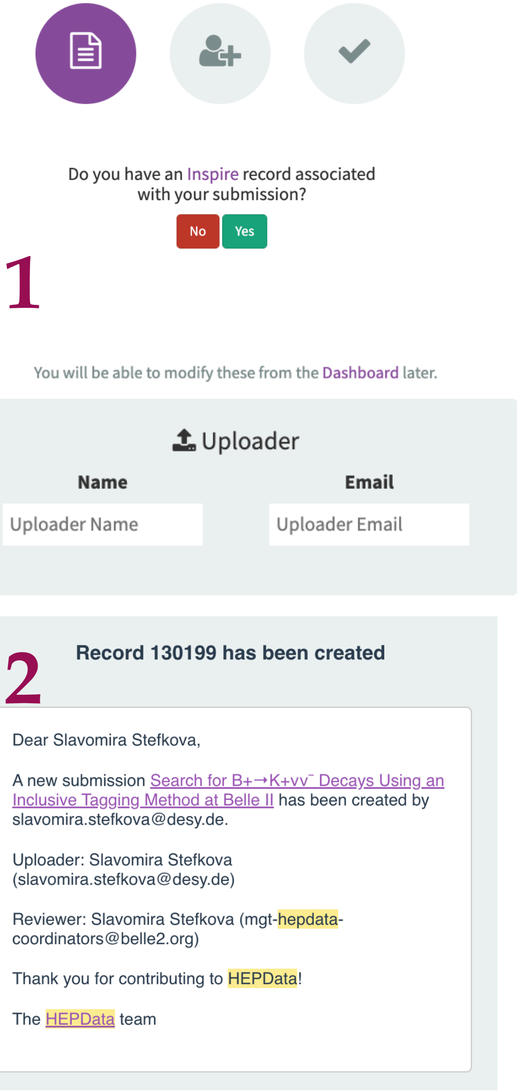
HEPData

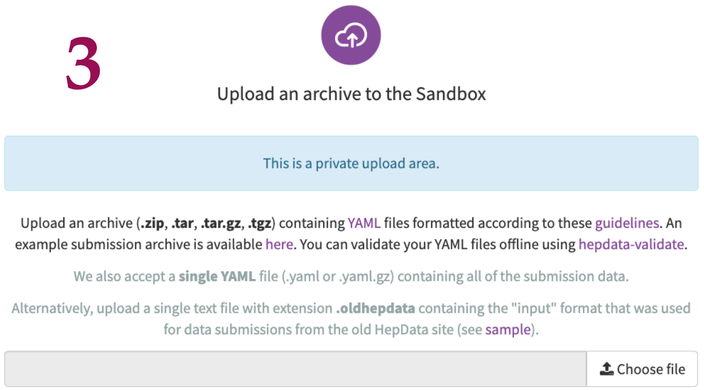
Give proper correlations

HEPData Submission Procedure

If you would like to make HEPData record:

1. Email mgt-hepdata-coordinators@belle2.org/alex.grecu@cern.ch to request a submission with following information:
   - **Inspire ID** of your record: e.g https://inspirehep.net/literature/**1860766**
   - Your name and email address
2. Coordinators will create a dashboard for you and you will be able to submit your record after receiving an automatised email from submissions@hepdata.net
3. Upload and submit your record
4. Coordinators will review your record
5. After successful review, coordinators will publish the HEPData record





HEPData Format

All details on submission formats: https://hepdata-submission.readthedocs.io/en/latest/

Submission should be done in `.zip,.tar,.tar.gz,.tgz` format and it should contain:

- Main file `submission.yaml` which tells about your entire submission:
  - content of your submission
  - what data files are in your submission
  - what they contain
  - any related material
  - keywords
- All data files which should also be in YAML or JSON format (more details can be found on https://hepdata-submission.readthedocs.io/en/latest/data_yaml.html)
- All associated figures in `.png` format

```yaml
name: "Table 1"
description: Describe the data.  The more you say, the easier it'll
keywords: # used for searching, possibly multiple values for each ke
- {name: reactions, values: [P P --> Z0 Z0 X]}
- {name: observables, values: [SIG]}
- {name: cmenergies, values: [7000.0]}
- {name: phrases, values: [Inclusive, Integrated Cross Section, Cros
data_file: data1.yaml
```

YAML data file example

```yaml
independent_variables:
- header: {name: Leading dilepton PT, units: GEV}
  values:
  - {low: 0, high: 60}
  - {low: 60, high: 100}
  - {low: 100, high: 200}
  - {low: 200, high: 600}
dependent_variables:
- header: {name: 10**6 * 1/SIG(fiducial) * D(SIG(fiducial))/DPT, uni
  qualifiers:
  - {name: RE, value: P P --> Z0 < LEPTON+ LEPTON- > Z0 < LEPTON+ LE
  - {name: SQRT(S), units: GEV, value: 7000}
  values:
  - value: 7000
    errors:
    - {symerror: 1100, label: stat}
    - {symerror: 79, label: 'sys,detector'}
    - {symerror: 15, label: 'sys,background'}
  - value: 9800
    errors:
    - {symerror: 1600, label: stat}
    - {symerror: 75, label: 'sys,detector'}
    - {symerror: 15, label: 'sys,background'}
  - value: 1600
    errors:
    - {symerror: 490, label: stat}
    - {symerror: 41, label: 'sys,detector'}
    - {symerror: 2, label: 'sys,background'}
  - value: 80
    errors:
    - {symerror: 60, label: stat}
    - {symerror: 2, label: 'sys,detector'}
    - {symerror: 0, label: 'sys,background'}
```

You can install hepdata-validator tool to check your submission!

NB: publication information such as the paper title, authors and abstract, or the journal reference and DOI, is pulled from the corresponding INSPIRE record

HEPData Format II

YAML data file example

```yaml
name: "Table 1"
description: Describe the data.  The more you say, the easier it'll
keywords: # used for searching, possibly multiple values for each ke
- {name: reactions, values: [P P --> Z0 Z0 X]}
- {name: observables, values: [SIG]}
- {name: cmenergies, values: [7000.0]}
- {name: phrases, values: [Inclusive, Integrated Cross Section, Cros
data_file: data1.yaml
```

```yaml
independent_variables:
- header: {name: Leading dilepton PT, units: GEV}
  values:
  - {low: 0, high: 60}
  - {low: 60, high: 100}
  - {low: 100, high: 200}
  - {low: 200, high: 600}
dependent_variables:
- header: {name: 10**6 * 1/SIG(fiducial) * D(SIG(fiducial))/DPT, uni
  qualifiers:
  - {name: RE, value: P P --> Z0 < LEPTON+ LEPTON- > Z0 < LEPTON+ LE
  - {name: SQRT(S), units: GEV, value: 7000}
  values:
  - value: 7000
    errors:
    - {symerror: 1100, label: stat}
    - {symerror: 79, label: 'sys,detector'}
    - {symerror: 15, label: 'sys,background'}
  - value: 9800
    errors:
    - {symerror: 1600, label: stat}
    - {symerror: 75, label: 'sys,detector'}
    - {symerror: 15, label: 'sys,background'}
  - value: 1600
```

Belle II Example: Search for $B^+ \to K^+ \nu\bar{\nu}$

[PRL 127, 181802 (2021)]

git clone ssh://git@stash.desy.de:7999/~sstefkov/hep_data_b_knunu.git
git clone https://github.com/Sally27/hep_data_b_knunu.git

HEPData Record for $B^+ \to K^+\nu\bar{\nu}$

[PRL 127, 181802 (2021)]

Browse all | Abudinén, F. et al.

Hide Publication Information

Search for B+→K+νν¯ Decays Using an Inclusive Tagging Method at Belle II

The Belle-II collaboration

Page 11: Resources

Journal | INSPIRE | Resources

Abstract (data abstract)

SuperKEKB Belle II. Measurement of the branching fraction of $B^+ \to K^+\nu\bar{\nu}$ at the Belle II experiment at the SuperKEKB. The analysed data sample corresponds to an integrated luminosity of 63 fb^{-1} collected at the $\Upsilon(4S)$ resonance and a sample of 9 fb^{-1} collected at an energy 60 MeV below the resonance between 2019-2021. Since no significant signal was observed, limit of 4.1×10^{-5} was set using CL$_s$ method.

Notify Participants

Page 14: Download

Download All

Page 12: Content

Table of contents

10.17182/hepdata.130199.v1/t1

------- Overview of HEPData Record ---------...

to be reviewed

Postfit yields Y(4S)

Figure 3 in https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.127.1...

10.17182/hepdata.130199.v1/t2

Yields in on-resonance data and as predicted by the simultaneous fit to the on- and off-resonance data, corresponding to an...

to be reviewed

Postfit yields off-resonance

Figure 1 in https://journals.aps.org/prl/supplemental/10.1103/PhysRe...

10.17182/hepdata.130199.v1/t3

Yields in off-resonance data and as predicted by the simultaneous fit to the on- and off-resonance data, corresponding to an...

to be reviewed

Expected and observed limit

Figure 2 in

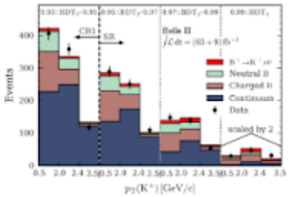
Page 13: Data Tables

Postfit yields Y(4S) 10.17182/hepdata.130199.v1/t2

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

Figure 3 in https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.127.181802

Yields in on-resonance data and as predicted by the simultaneous fit to the on- and off-resonance data, corresponding to an integrated luminosity of 63 and 9 fb^{-1}, respectively. The predicted yields are shown individually for charged and neutral B-meson decays and the five continuum background categories. The leftmost three bins belong to the first control region (CR1) with BDT$_2 \in [0.93; 0.95]$ and the other nine bins correspond to the signal region (SR), three for each range of BDT$_2 \in [0.95; 0.97; 0.99; 1.0]$. Each set of three bins is defined by $p_T(K^+) \in [0.5; 2.0; 2.4; 3.5]\mathrm{GeV}/c^2$.



observables

signal strenghth μ

phrases

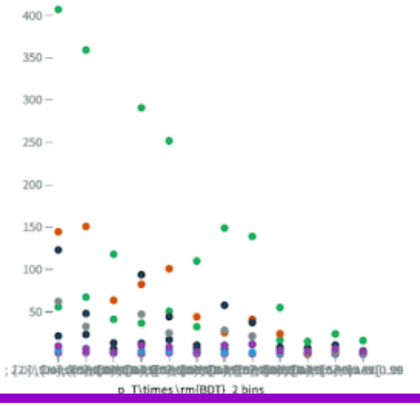
FCNC | b --> s l l transition | electroweak penguin decay | missing energy

reactions

$B^+ \to K^+\nu\bar{\nu}$

	yield				
$p_T \times \mathrm{BDT}_2$ bins	Observed data	Number of signal events $B^+ \to K^+\nu\bar{\nu}$	Number of events from charged B backgrounds	Number of events from neutral B backgrounds	Number of events from $c\bar{c}$ backgrounds
$[0.5; 2.0] \times [0.93; 0.95]$	407.0	9.6767794	123.073966	62.1993461	144.517922
$[2.0; 2.4] \times [0.93; 0.95]$	359.0	6.32168858	47.9504222	32.7334206	150.685436
$[2.4; 3.5] \times [0.93; 0.95]$	118.0	1.6782059	6.20851002	3.56989512	63.613553
$[0.5; 2.0] \times [0.95; 0.97]$	291.0	10.2102786	93.8582929	47.179443	82.5892068
$[2.0; 2.4] \times [0.95; 0.97]$	252.0	8.34518351	44.2392619	25.0057923	101.007991

Visualize



Resources for $B^+ \to K^+\nu\bar{\nu}$

[PRL 127, 181802 (2021)]

Resources (specified in `submission.yaml`):

- List of all additional resources, e.g:
  - Link to supplemental material
  - Home page for the analysis
  - Histfactory / pyhf template
  - …..what you deem important

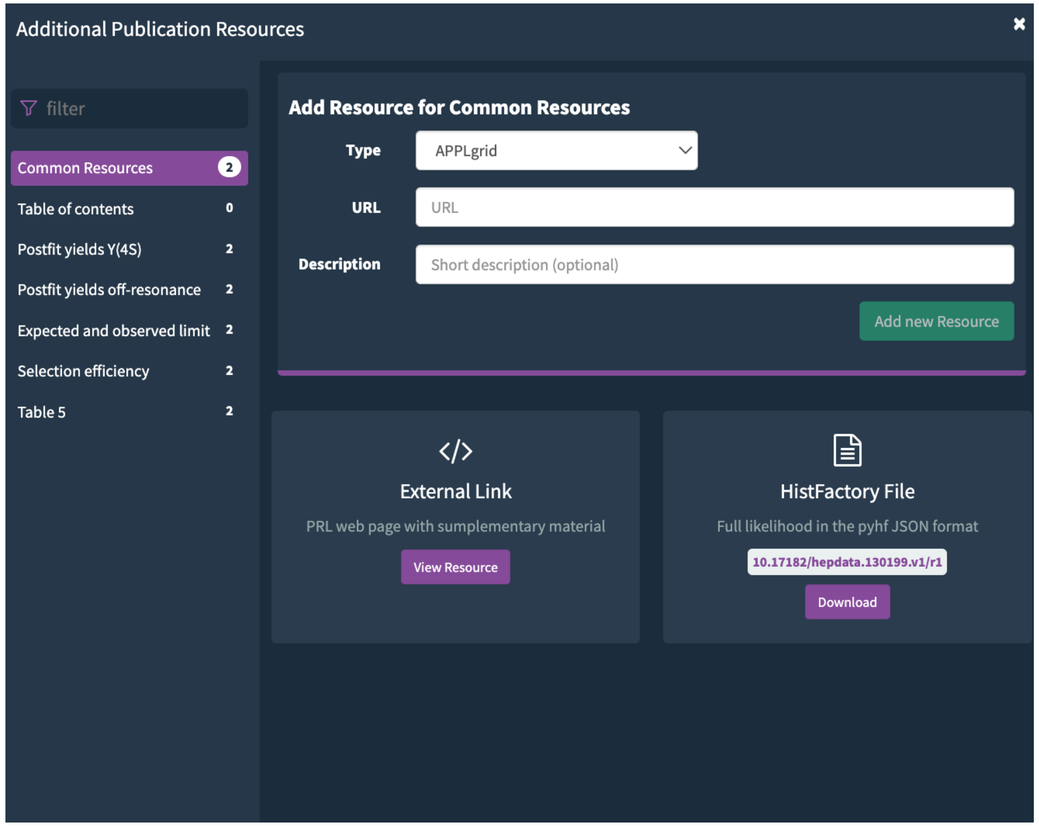
Phys.Rev.Lett. 127 (2021) 181802, 2021.

https://doi.org/10.17182/hepdata.130199

Journal | INSPIRE | Resources

Abstract (data abstract)

SuperKEKB Belle II. Measurement of the branching fraction of $B^+ \to K^+\nu\bar{\nu}$ at the Belle II experiment at the SuperKEKB. The analysed data sample corresponds to an integrated luminosity of 63 fb^{-1} collected at the $\Upsilon(4S)$ resonance and a sample of 9 fb^{-1} collected at an energy 60 MeV below the resonance between 2019-2021. Since no significant signal was observed, limit of 4.1×10^{-5} was set using CL$_s$ method.



Content of $B^+ \to K^+\nu\bar{\nu}$

[PRL 127, 181802 (2021)]

Content (specified in `submission.yaml`):

- Describe the content of HEPData record
- Provide clickable links to the data entries for faster access

Table of contents 10.17182/hepdata.130199.v1/t1

- - - - - - - - Overview of HEPData Record - - - - - - - -

Post-fit yields:

- Y(4S)
- Off-resonance

Exclusion limit:

- Expected limit and observed limit

Efficiency:

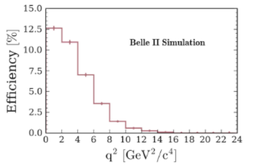
- Selection efficiency as a function of q^2

Selection efficiency 10.17182/hepdata.130199.v1/t5

Resources | https://www.hepdata.net | JSC

Figure 3 in https://journals.aps.org/prl/supplemental/10.1103/PhysRevLett.127.181802/suppl_mat.pdf

Signal efficiency as a function of the dineutrino invariant mass squared q^2 for events in the signal region (SR) ($\mathrm{BDT}_1 > 0.9$ and $\mathrm{BDT}_2 > 0.95$). The error bars indicate the statistical uncertainty.



phrases

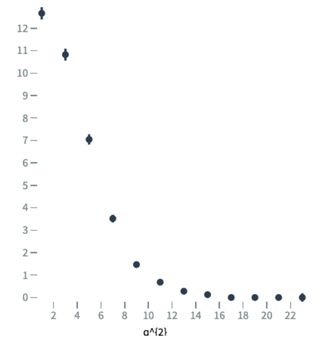
FCNC | b --> s l l transition | electroweak penguin decay | missing energy

reactions

$B^+ \to K^+\nu\bar{\nu}$

Luminosity	63+9 fb^{-1}
q^2	**Efficiency**
1.0	12.66745696 ±0.27207295
3.0	10.82571692 ±0.26463688
5.0	7.04488885 ±0.2278063
7.0	3.51769225 ±0.1711566
9.0	1.46683133 ±0.11813559
11.0	0.68175914 ±0.08670158
13.0	0.27954132 ±0.06074916

Visualize



Data Tables in $B^+ \to K^+\nu\bar{\nu}$

[PRL 127, 181802 (2021)]

Examples of data tables (specified in `data1.yaml`, `data2.yaml`, …):

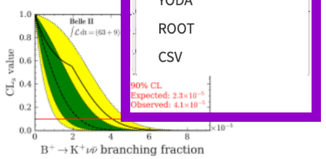
1. With CL_s limit table from $B^+ \to K^+\nu\bar{\nu}$ you can access 90 and/or 95% CL value for BF
2. With selection efficiency table from $B^+ \to K^+\nu\bar{\nu}$ you can do simplified NP recast

Expected and observed limit 10.17182/hepdata.130199.v1/t4

Resources | https://www.hepdata.net | JSON | YAML | YODA | ROOT | CSV

Figure 2 in https://journals.aps.org/prl/supplemental/10.1103/PhysRevLett.127.181802/suppl_mat.pdf

CL_s value as a function of the branching fraction of $B^+ \to K^+\nu\bar{\nu}$ for expected and observed signal yields and the corresponding upper limits at 90% confidence level (CL). The expected limit is derived for the background-only hypothesis. The observed limit is derived from a simultaneous fit to the on-resonance and off-resonance data, corresponding to an integrated luminosity of 63 fb^{-1} and 9 fb^{-1}, respectively.



1

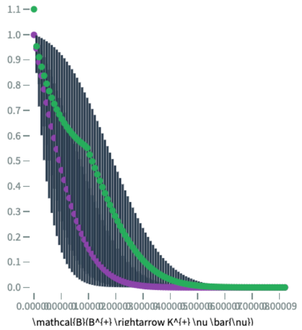
phrases: FCNC | b --> s l l transition | electroweak penguin decay | missing energy

reactions: $B^+ \to K^+\nu\bar{\nu}$

Showing 50 of 100 values — Show All 100 values

Limit	Observed	Expected
Luminosity	63+9 fb^{-1}	
$\mathcal{B}(B^+ \to K^+\nu\bar{\nu})$	**CL_s value**	
0.0	1.1	0.998621089 +0.0008814489999999786 -0.0012545640000000136 1 sigma +0.0012832659999999496 -0.0027154190000000052 2 sigma
9.292929292929293e-07	0.953245962	0.945208474 +0.034347557000000003 -0.046424566999999997 1 sigma +0.05072460500000009 -0.097479443 2 sigma
1.8585858585858586e-06	0.910805129	0.890664314 +0.06709263499999996 -0.085949371 1 sigma +0.10062385900000004 -0.17505751300000005 2 sigma
2.787878787878788e-06	0.872237598	0.836616884 +0.09800288000000001 -0.11892367299999995 1 sigma +0.14939054699999998 -0.23511698599999997 2 sigma
3.7171717171717173e-06	0.837177085	0.783342012 +0.12684882100000006 -0.14572960699999993 1 sigma +0.19669320499999998 -0.279873672 2 sigma

Visualize

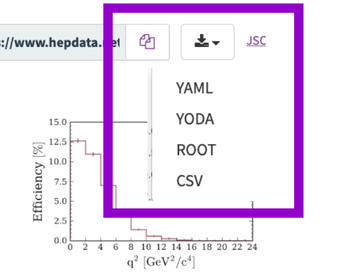


Selection efficiency 10.17182/hepdata.130199.v1/t5

Resources | https://www.hepdata.net | JSC | YAML | YODA | ROOT | CSV

Figure 3 in https://journals.aps.org/prl/supplemental/10.1103/PhysRevLett.127.181802/suppl_mat.pdf

Signal efficiency as a function of the dineutrino invariant mass squared q^2 for events in the signal region (SR) ($\mathrm{BDT}_1 > 0.9$ and $\mathrm{BDT}_2 > 0.95$). The error bars indicate the statistical uncertainty.



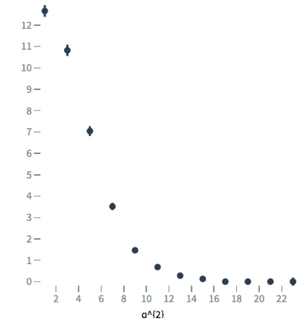
2

phrases: FCNC | b --> s l l transition | electroweak penguin decay | missing energy

reactions: $B^+ \to K^+\nu\bar{\nu}$

Luminosity	63+9 fb^{-1}
q^2	**Efficiency**
1.0	12.66745696 ±0.27207295
3.0	10.82571692 ±0.26463688
5.0	7.04488885 ±0.2278063
7.0	3.51769225 ±0.1711566
9.0	1.46683133 ±0.11813559
11.0	0.68175914 ±0.08670158
13.0	0.27954132 ±0.06074916

Visualize

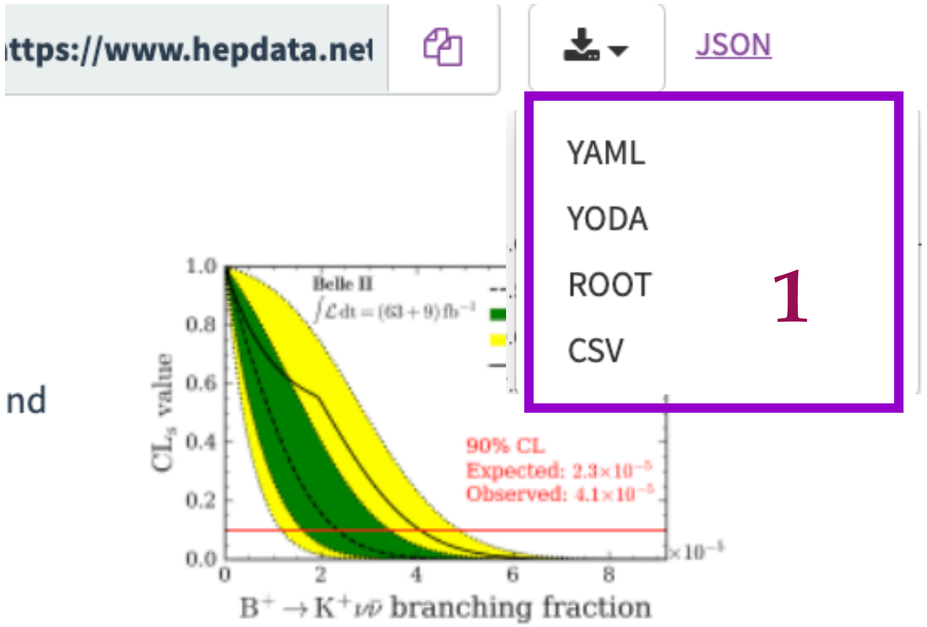


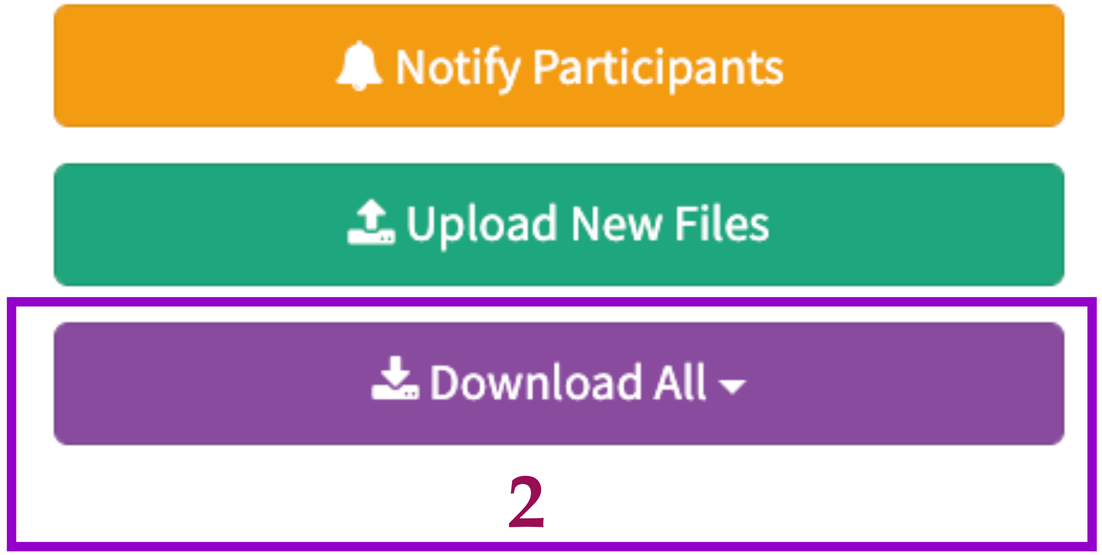
Download Data from $B^+ \to K^+ \nu\bar{\nu}$

[PRL 127, 181802 (2021)]

Download options:

1. Individual Tables
2. Full HEPData Record





What to publish?

What to Publish: 1D

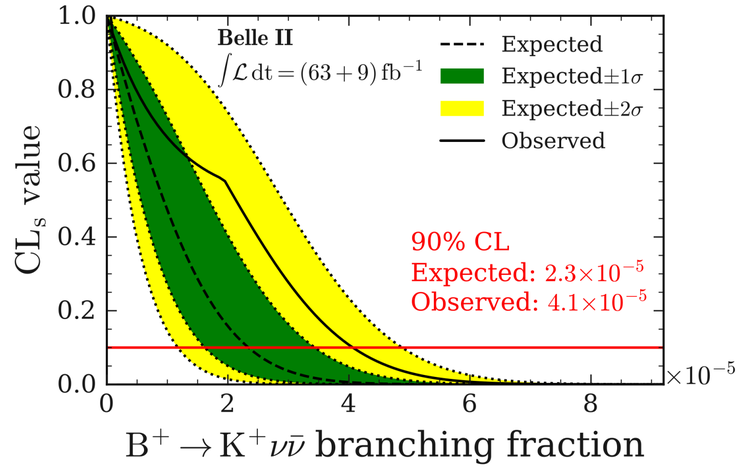
Motto: The more the better :)

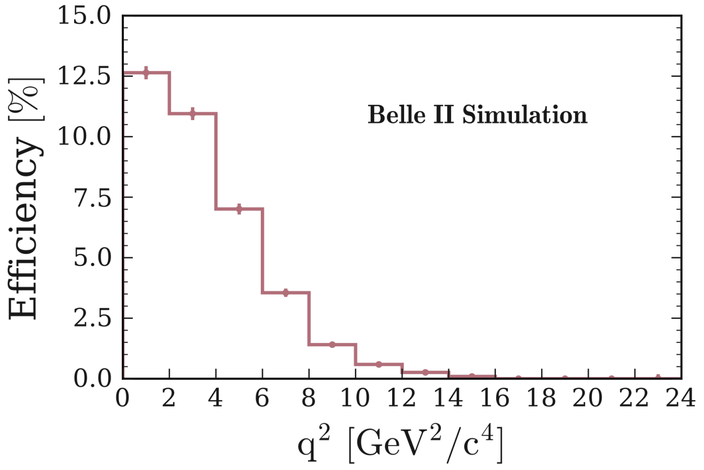
What we can publish in HEPData for 1D results:

- Likelihoods (e.g pyhf/histfactory format for binned fit)
- Generator and selection efficiency as a function of recasting variables
- Central values with uncertainties
- Expected and observed exclusion limits
- Rivet routines

……. basically all the plots + likelihood


```
"channels": [
    {
        "name": "A__B_sig_K_pt_vs_bdt_v24_ff_weights_T09_zoomSR_Y4S",
        "samples": [
            {
                "name": "signal",
                "data": [2.391054, 1.966424, 0.559938, 2.43141, 2.775106, 0.90273, 0.75645, 1.427192, 0.594811],
                "modifiers": [
                    {
                        "name": "uncorr_siguncrt_signal_zoomSR_Y4S",
                        "type": "staterror",
                        "data": [0.013203, 0.011053, 0.005478, 0.013141, 0.013078, 0.007, 0.007286, 0.009359, 0.005657]
                    },
                    {
                        "name": "mu",
                        "type": "normfactor",
                        "data": null
                    },
```



What to Publish: 2D

What we can publish in HEPData for 2D results:

- 2D joint likelihood
- Generator and selection efficiency as a function of recasting variables
- Correlation matrix (likelihood contours) for measured observables (e.g axial vs scalar BF) + profiled 1D projections
- Correlation matrix (likelihood contours) for the NP parameters

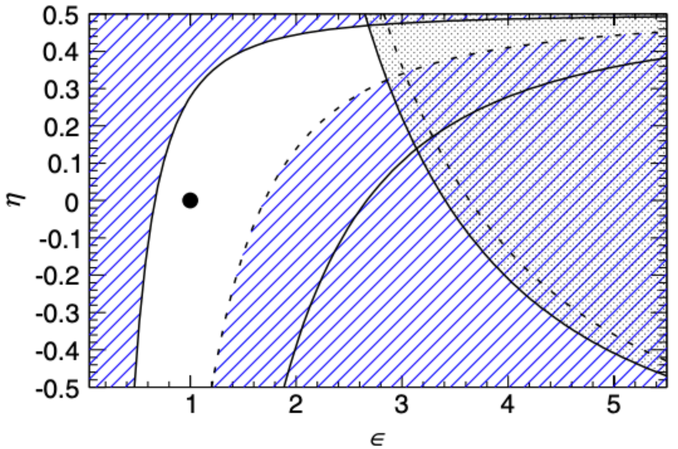
Involves flavio

[PRD 87, 112005 (2013)](#)

BaBar $B^+ \to K^+ \nu\bar{\nu}$

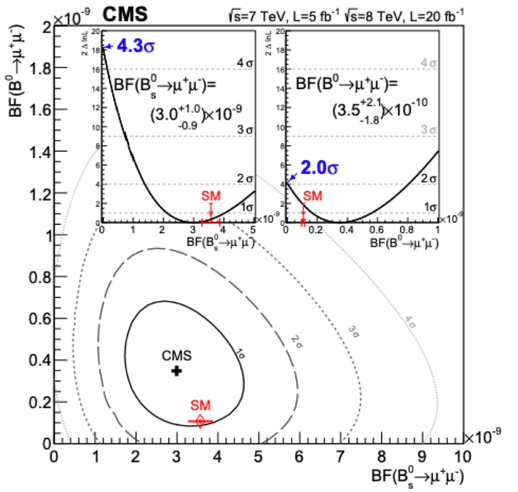
$$\epsilon \equiv \frac{\sqrt{|C_L^\nu|^2 + |C_R^\nu|^2}}{|C_{L,\mathrm{SM}}^\nu|}\,, \quad \eta \equiv \frac{-\mathrm{Re}(C_L^\nu C_R^{\nu *})}{|C_L^\nu|^2 + |C_R^\nu|^2}$$

$$\mathrm{BR}(B \to K^* \nu\bar{\nu}) = 6.8 \times 10^{-6}\,(1 + 1.31\,\eta)\epsilon^2\,,$$
$$\mathrm{BR}(B \to K \nu\bar{\nu}) = 4.5 \times 10^{-6}\,(1 - 2\,\eta)\epsilon^2\,,$$



[PRL 111 101804 (2013)](#)

CMS and LHCb $B_s^0 \to \mu^+\mu^-$



What to Publish: nD

What is the best is to published in HEPData:

- nD joint likelihood
- Generator and selection efficiencies
- Correlation matrices

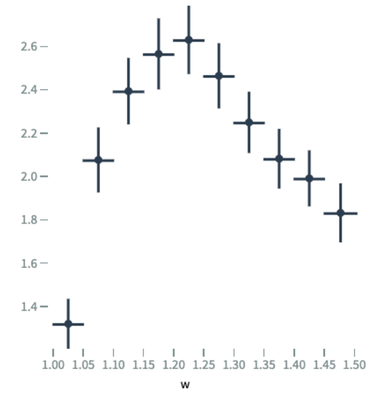
But the main guidance is your physics case, so please ask your theory friends about what is necessary/helpful for them!

Belle Example

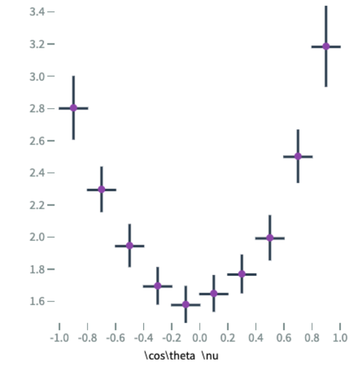
Precise determination of the CKM matrix element $|V_{cb}|$ with $\bar{B}^0 \to D^{*+} \ell^- \bar{\nu}_\ell$ decays with hadronic tagging at Belle

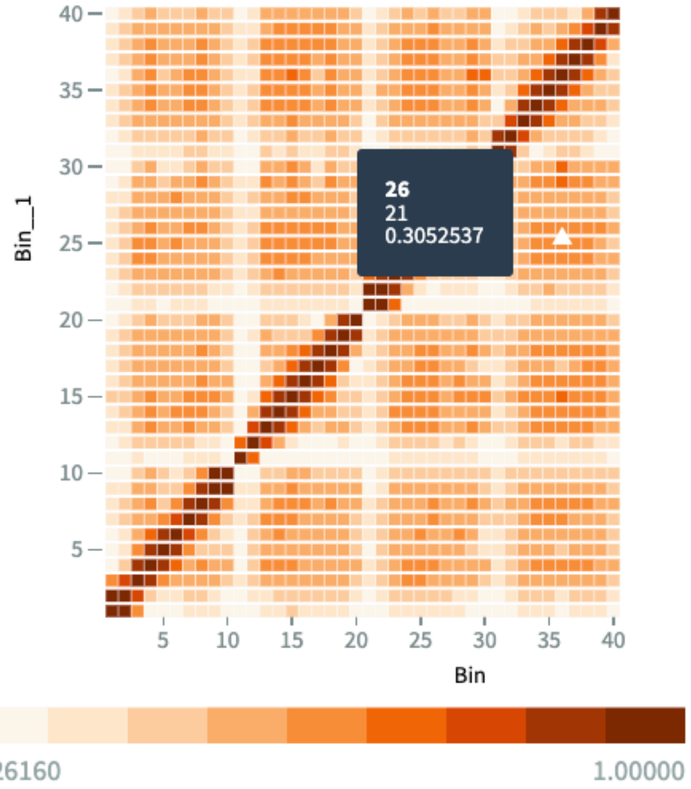
The correlation matrix of the unfolded differential rates. The full error covariance can be obtained by combining the quoted error in the other tables with these values. The ordering of the correlations is $\{w, \cos\theta_\nu, \cos\theta_\ell, \chi\}$, where each variable has 10 bins.

Visualize



Visualize





Rivet

https://rivet.hepforge.org/

Rivet is an analysis toolkit

- Short for “Robust Independent Validation of Experiment and Theory”
- Framework in which one can provide complicated selections or observables to theory colleagues.

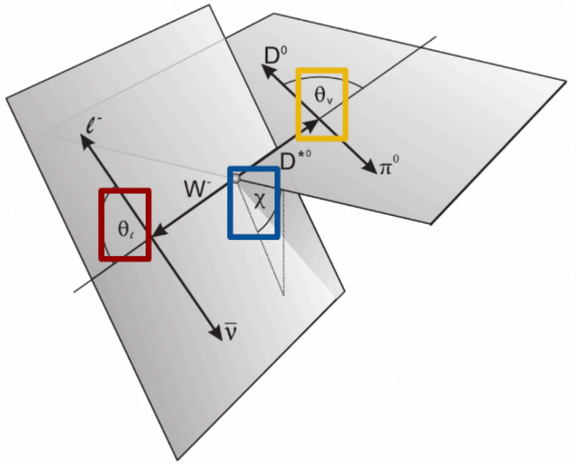
Example use case:

- You measured a complicated angular observable and want your theory colleagues to use the same definition when making interpretations or predictions for your observable.
- You want to allow theorists to make a reinterpretation of your limit and document what cuts they would need to apply to approximate your experimental efficiencies

Belle Example

```
// This is the angle analysis
if (foundDdecay) {

  // First boost all relevant momenta into the B-rest frame
  const LorentzTransform B_boost = LorentzTransform::mkFrameTransformFromBeta(pB.betaVec());
  // Momenta in B rest frame:
  FourMomentum lv_brest_Dstar = B_boost.transform(pDs);//lab2brest(gp_Dstar.particle.p());
  FourMomentum lv_brest_w     = B_boost.transform(pB - pDs); //lab2brest(p_lv_w);
  FourMomentum lv_brest_D     = B_boost.transform(pD); //lab2brest(gp_D.particle.p());
  FourMomentum lv_brest_lep   = B_boost.transform(pl); //lab2brest(gp_lep.p());

  const LorentzTransform Ds_boost = LorentzTransform::mkFrameTransformFromBeta(pDs.betaVec());
  FourMomentum lv_Dstarrest_D     = Ds_boost.transform(lv_brest_D);
  const LorentzTransform W_boost  = LorentzTransform::mkFrameTransformFromBeta((pB-pDs).betaVec(
  FourMomentum lv_wrest_lep       = W_boost.transform(lv_brest_lep);
```

$$S_5(q^2) = \left(\frac{d\Gamma}{dq^2}\right)^{-1} \left[\int_0^{\pi/2} - \int_{\pi/2}^{\pi} - \int_{\pi}^{3\pi/2} + \int_{3\pi/2}^{2\pi}\right] d\chi \left[\int_0^1 - \int_{-1}^0\right] d\cos\theta^* \frac{d^3\Gamma}{dq^2 d\cos\theta^* d\chi},$$



Conclusion

This presentation was just a very quick guide-through of what HEPData is, I hope I convinced you that it is very important to publish all available data

Please contact your local coordinator if you want to make HEPData submission

Snakemake

Workflow Management System

As the analyses get complicated the workflow management system is important!

Workflow Management System

As the analyses get complicated the workflow management system is important!

1. Luigi:

- Belle II's **b2luigi**:
  - https://www-kseta.ttp.kit.edu/fellows/Felix.Metzner/talks/b2luigi_introduction.pdf ,
  - https://b2luigi.readthedocs.io/en/stable/advanced/basf2-examples.html,
  - https://software.belle2.org/development/sphinx/online_book/workflowmanagement/b2luigi.html?highlight=b2luigi
- **LAW:**
  - https://indico.cern.ch/event/1102574/contributions/4638656/attachments/2361414/4031168/2021-12-08_hsf_law.pdf,
  - https://github.com/riga/law

2. Snakemake:

- Belle II: https://software.belle2.org/development/sphinx/online_book/workflowmanagement/snakemake.html
- LHCb: https://indico.cern.ch/event/1102574/contributions/4638655/attachments/2361381/4031133/snakemake%20workflow%20manager.pdf
- HSF: https://hsf-training.github.io/analysis-essentials/snakemake/README.html

Workflow Management System

As the analyses get complicated the workflow management system is important!

1. Luigi:

- Belle II's **b2luigi**:
  - https://www-kseta.ttp.kit.edu/fellows/Felix.Metzner/talks/b2luigi_introduction.pdf ,
  - https://b2luigi.readthedocs.io/en/stable/advanced/basf2-examples.html,
  - https://software.belle2.org/development/sphinx/online_book/workflowmanagement/b2luigi.html?highlight=b2luigi
- **LAW:**
  - https://indico.cern.ch/event/1102574/contributions/4638656/attachments/2361414/4031168/2021-12-08_hsf_law.pdf,
  - https://github.com/riga/law

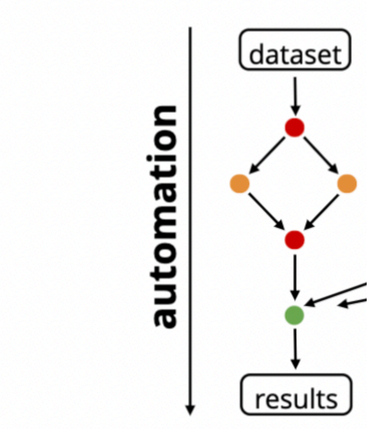
2. Snakemake:

- Belle II: https://software.belle2.org/development/sphinx/online_book/workflowmanagement/snakemake.html
- LHCb: https://indico.cern.ch/event/1102574/contributions/4638655/attachments/2361381/4031133/snakemake%20workflow%20manager.pdf
- HSF: https://hsf-training.github.io/analysis-essentials/snakemake/README.html

Snakemake

Snakemake is:

- Workflow management system that
  - performs code execution from raw data to final figures:
    - **Document** parameters, tools, versions
    - **Execute** without manual intervention

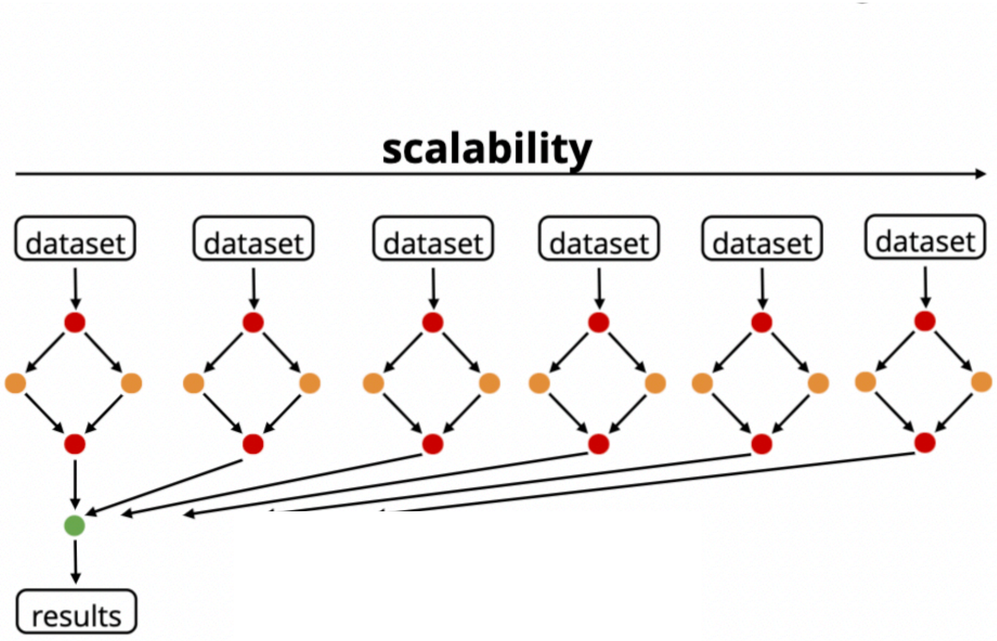


snakemake

Snakemake

Snakemake is:

- Workflow management system that
  - performs code execution from raw data to final figures:
    - **document** parameters, tools, versions
    - **execute** without manual intervention
  - handles paralellisation efficiently:
    - applicable to large datasets
    - on different computing platforms

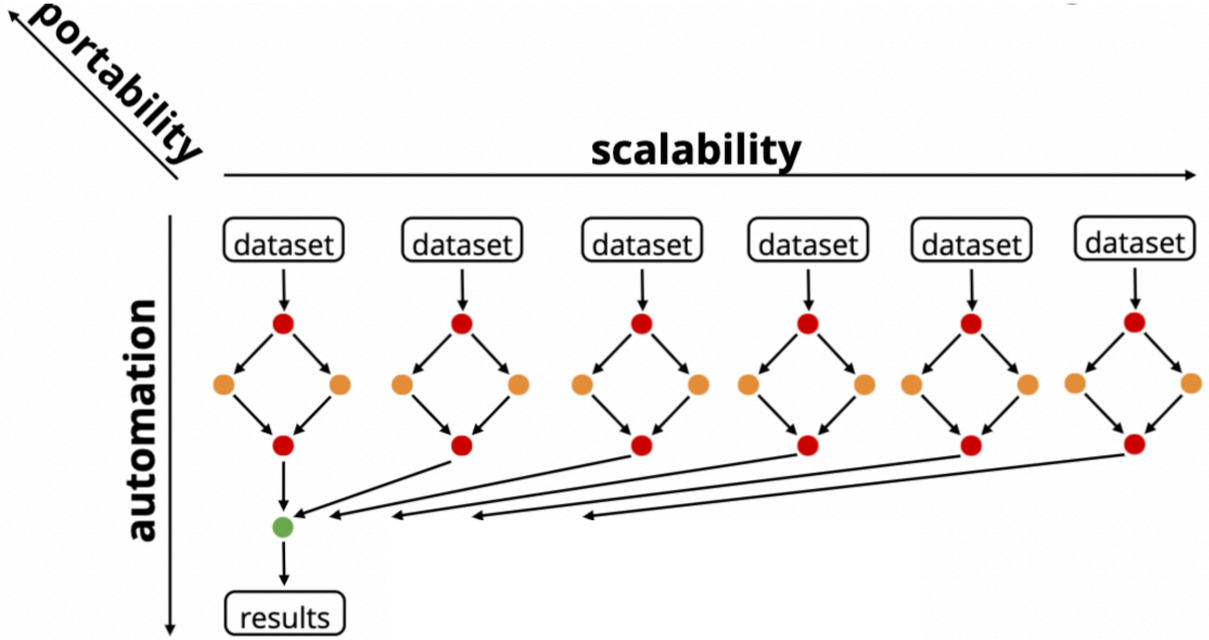


snakemake

Snakemake

Snakemake is:

- Workflow management system that
  - performs code execution from raw data to final figures:
    - **document** parameters, tools, versions
    - **execute** without manual intervention
  - handles paralellisation efficiently:
    - applicable to large datasets
    - on different computing platforms
  - is deployable easily on a different system/ platform/infrastructure



snakemake

Snakemake

Why to use  ?

Separation from your analysis code	Easy to add to your analysis
Built on top of python	Easy to learn + make your own python wrappers
Can be run anywhere	Supports batch jobs, easy parallelization
Preservation is easier	Supports docker, conda and allows to tag versions
Community	Forums, supports, stackexchange
Utilities	CWL tools, XRootD, paramspace

HTCondor High Throughput Computing

CONDA

COMMON WORKFLOW LANGUAGE

*Taken from https://indico.cern.ch/event/1102574/contributions/4638655/attachments/2361381/4031133/snakemake%20workflow%20manager.pdf

Snakemake

Popularity:

- Use beyond particle physics
- > 1300 downloads

Languages:

- Support for many programming languages including python and bash

Snakemake

Popularity:

- Use beyond particle physics
- > 1300 downloads

Languages:

- Support for many programming languages including python and bash

Now we will install snakelike and follow an easy example

Snakemake

Instalation guidelines :

1. `conda install -n base -c conda-forge mamba`
   (install a Conda-based Python3 distribution)
2. `conda activate base`
   (activate the environment)
3. `mamba create -c conda-forge -c bioconda -n snakemake`
   (this will install snakemake into an isolated software environment)
4. `conda activate snakemake`
   (activate the environment)
5. `snakemake –help`
   (Look at your options)
6. **…**
   (Work)
7. `deactivate`
   (Deactivate the environment)

Snakemake tutorial

We will follow basic tutorial from HSF!

- Let's create our working directory and change to that directory
- Let's create input directory with 26 input files: `a.in, b.in, …, z.in`
- Open `Snakefile` in your favourite text editor

```
mkdir basic_tutorial
cd basic_tutorial
mkdir input
touch input/{a..z}.in
ls input/

vim Snakefile
```

Snakemake Logic

Workflow Logic Code:

- Encoded in **`Snakefile`**
- Processing steps are called **`rules`** written in a simple Python-based language
- Backbone of a **`rule`** is:
  - **`Input:`** files
  - **`Output:`** files and directive to create latter
  - **`Shell:`** commands, python code in **`run:`** or a **`script:`** path

```
rule my_first_rule:
     input: "my_input_1", "my_input_2"
     output: "my_output",
     shell: "Some command to go from in to out"
```

Snakefile

Snakemake Logic

Workflow Logic Actual Code:

- Encoded in **`Snakefile`**
- Processing steps are called **`rules`** written in a simple Python-based language
- Backbone of a **`rule`** is:
  - **`Input:`** files
  - **`Output:`** files and directive to create latter
  - **`Shell:`** commands, python code in **`run:`** or a **`script:`** path

```
rule name_files:
     input: "input/a.in",
     output: "output/a.out",
     shell: "cat input/a.in > output/a.out && echo 'My name is a.' >>
     output/a.out"
```

Snakefile

Execute Snakefile

Execute your first rule using 1 core:

- Snakemake knows how to creat `output/a.out`

```
$ snakemake output/a.out --cores 1
```

Execute Snakefile

Execute your first rule using 1 core:

- Snakemake knows how to create `output/a.out` including output directory

```
$ snakemake output/a.out --cores 1
```

```
Building DAG of jobs...
Using shell: /cvmfs/lhcbdev.cern.ch/conda/envs/default/2021-09-07_04-06/linux-64/bin/bash
Provided cores: 1 (use --cores to define parallelism)
Rules claiming more threads will be scaled down.
Job stats:
job           count    min threads    max threads
----------  -------  -------------  -------------
name_files        1              1              1
total             1              1              1

Select jobs to execute...

[Mon Nov 15 18:30:25 2021]
rule name_files:
    input: input/a.in
    output: output/a.out
    jobid: 0
    resources: tmpdir=/tmp/username

[Mon Nov 15 18:30:26 2021]
Finished job 0.
1 of 1 steps (100%) done
Complete log: /afs/cern.ch/user/u/username/basic_tutorial/.snakemake/log/2021-11-15T183022.449
```

Execute Snakefile

Execute your first rule using 1 core:

- Snakemake knows how to create `output/a.out`

```
$ snakemake output/a.out --cores 1
```

- Test with dry run

```
$ snakemake output/a.out --cores 1 -n
```

What happens if you run snakemake output/a.out --cores 1 again?

Execute Snakefile

Execute your first rule using 1 core:

- Snakemake knows how to create `output/a.out`

```
$ snakemake output/a.out --cores 1
```

- Test with dry run

```
$ snakemake output/a.out --cores 1 -n
```

- Force rerunning

```
$ snakemake output/a.out --cores 1 -F
```

Snakemake Logic

What about the 25 other rules with files to create?

- Take a moment to understand what is happening in this rule declaration.
- Wildcards are defined in `output` and can then be accessed in `input` and `shell`.
- **All wildcards must be present in `output`**
- We are essentially telling `Snakemake` that this is a rule that can create files of the form `output/{name}.out`; we cannot specify wildcards in `input` that are not present in `output` because Snakemake will not know what they should be
- Did you notice something special about wildcards?

```
rule name_files:
    input: "input/{name}.in",
    output: "output/{name}.out",
    shell: "cat {input} > {output} && echo 'My name is {wildcards.name}'
    >> {output}"
```

Snakemake Logic

What about the 25 other rules with files to create?

- Take a moment to understand what is happening in this rule declaration.
- Wildcards are defined in `output` and can then be accessed in `input` and `shell`.
- **All wildcards must be present in `output`**
- We are essentially telling Snakemake that this is a rule that can create files of the form `output/{name}.out`; we cannot specify wildcards in `input` that are not present in `output` because Snakemake will not know what they should be
- Did you notice something special about wildcards?

```
rule name_files:
    input: "input/{name}.in",
    output: "output/{name}.out",
    shell: "cat {input} > {output} && echo 'My name is
    {wildcards.name}' >> {output}"
```

- Let's run on 4 cores; what is different?

```
$ snakemake output/{a..z}.out –cores 4
```

Snakemake Logic

We can also create a rule to tell Snakefile which files to create

- Let's name this rule `name_all` , declare it at the beginning with input but no output
- Snakemake will perform the first rule as default rule
- Notice that list comprehension is done in `python`

```
rule name_all::
     input: [f"output/{chr(x)}.out" for x in range(ord('a'), ord('z') + 1)]

rule name_files:
     input: "input/{name}.in",
     output: "output/{name}.out",
     shell: "cat {input} > {output} && echo 'My name is {wildcards.name}' >>
     {output}"
```

```
$ rm -r output/

$ snakemake –cores 4
```

Snakemake Logic

We can also create a rule to tell Snakefile which files to create

- Let's name this rule `name_all` , declare it at the beginning with input but no output
- Snakemake will perform the first rule as default rule
- Notice that list comprehension is done in `python`

```
rule name_all::
     input: [f"output/{chr(x)}.out" for x in range(ord('a'), ord('z') + 1)]

rule name_files:
     input: "input/{name}.in",
     output: "output/{name}.out",
     shell: "cat {input} > {output} && echo 'My name is {wildcards.name}' >>
     {output}"
```

```
$ rm -r output/

$ snakemake —cores 4
```

- What about creating `z.foo`? Is it possible?

Advanced tutorial

https://hsf-training.github.io/analysis-essentials/snakemake/README.html

```
mkdir advanced_tutorial
cd advanced_tutorial
wget https://github.com/hsf-training/analysis-essentials/raw/master/snakemake/
code/advanced_tutorial/input.tar
tar -xvf input.tar
ls input/

vim Snakefile
```

Task 1: Write a Snakefile with a single rule that creates an output file with Luca's address and phone number.

Advanced tutorial

https://hsf-training.github.io/analysis-essentials/snakemake/README.html

```
mkdir advanced_tutorial
cd advanced_tutorial
wget https://github.com/hsf-training/analysis-essentials/raw/master/snakemake/
code/advanced_tutorial/input.tar
tar -xvf input.tar
ls input/

vim Snakefile
```

Task 1: Write a Snakefile with a single rule that creates an output file with Luca's address and phone number.
Hint: the command grep lists all lines in a file containing a given text!

Advanced tutorial

Task 1: Write a Snakefile with a single rule that creates an output file with Luca's address and phone number.
Hint: the command grep lists all lines in a file containing a given text.

```
rule Luca_info:
    input: "input/address.txt", "input/phone.txt"
    output: "output/Luca/info.txt"
    shell: "grep Luca {input[0]} > {output} && grep Luca {input[1]} >> {output}"
```

Or

```
rule Luca_info:
    input:
        address="input/address.txt",
        phone="input/phone.txt",
    output: "output/Luca/info.txt"
    shell: "grep Luca {input.adress} > {output} && grep Luca {input.phone} >> {output}"
```

Advanced tutorial

```
rule merge_data:
    input:
        address="output/Luca/address.txt",
        phone="output/Luca/phone.txt",
    output: "output/Luca/info.txt"
    shell:"cat {input.address} > {output} && cat {input.phone} >> {output}"


rule get_address:
    input: "input/address.txt"
    output: "output/Luca/address.txt"
    shell: "grep Luca {input} > {output}"



rule get_phone:
    input: "input/phone.txt"
    output: “output/Luca/phone.txt"
    shell: "grep Luca {input} > {output}"
```

Task 2: Run in parallel getting the phone and address information

Advanced tutorial

```
rule merge_data:
    input:
        address="output/Luca/address.txt",
        phone="output/Luca/phone.txt",
    output: "output/Luca/info.txt"
    shell:"cat {input.address} > {output} && cat {input.phone} >> {output}"


rule get_address:
    input: "input/address.txt"
    output: temp("output/Luca/address.txt")
    shell: "grep Luca {input} > {output}"



rule get_phone:
    input: "input/phone.txt"
    output: temp("output/Luca/phone.txt")
    shell: "grep Luca {input} > {output}"
```

Task 3: We do not need the intermediate files, let's use `temp` function

Advanced tutorial

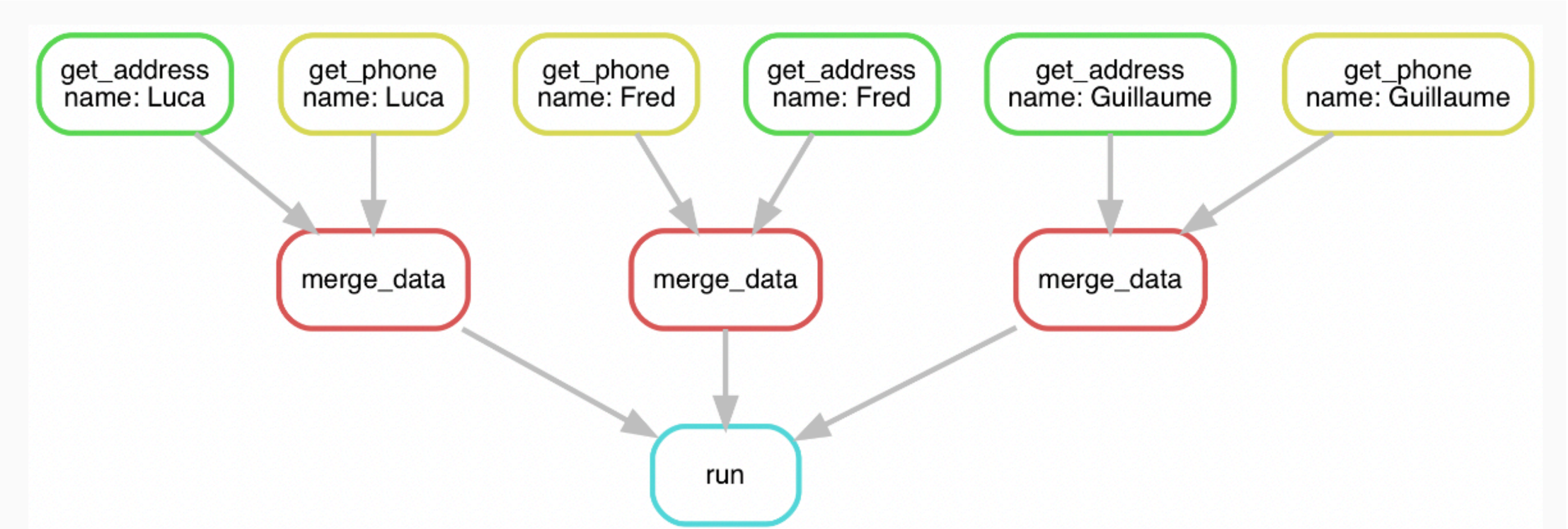
Task 4. Let's extend the program to other names: Fred, Guillaume

```
rule run:
    input: ["output/Luca/info.txt", "output/Fred/info.txt", “output/Guillaume/info.txt"]

rule merge_data:
    input:
        address="output/{name}/address.txt",
        phone="output/{name}/phone.txt",
    output: "output/{name}/info.txt"
    shell:"cat {input.address} > {output} && cat {input.phone} >> {output}"
```

```
rule get_address:
    input: "input/address.txt"
    output: temp("output/Luca/address.txt")
    shell: "grep {wildcards.name} {input} > {output}”

rule get_phone:
    input: "input/phone.txt"
    output: temp("output/Luca/phone.txt")
    shell: "grep {wildcards.name} {input} > {output}"
```

Advanced tutorial: DAG

Snakemake logic can be visualised in Directed Acyclic Graph (DAG)



Directed acyclic graph of the tasks until now

Advanced tutorial: More wildcards

Task 4. Let's merge get_address and get_info since they are very similar

```
rule run:
    input: ["output/Luca/info.txt", "output/Fred/info.txt", “output/Guillaume/info.txt"]

rule merge_data:
    input:
        address="output/{name}/address.txt",
        phone="output/{name}/phone.txt",
    output: "output/{name}/info.txt"
    shell:"cat {input.address} > {output} && cat {input.phone} >> {output}"
```

```
rule get_info:
    input: "input/{info}.txt"
    output: temp("output/Luca/{info}.txt")
    shell: "grep {wildcards.name} {input} > {output}"
```

Advanced tutorial: use python script

```python
"""Extract information about a person."""
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("name", help="person of interest")
parser.add_argument("infile", help="text file with information")
parser.add_argument("outfile", help="where to write extracted
information")
args = parser.parse_args()

# -- find information
found = None
with open(args.infile) as f:
    for ln in f:
        if ln.startswith(args.name):
            found = ln
            break  # assume only one line with info
info = found[len(args.name):]
# -- write information
with open(args.outfile, "w") as f:
    f.write(info)
```

Task 5. Write `get_info.py` and use that as an argument in `Snakefile`

Advanced tutorial: use python script

Task 6. Use is as executable in the workflow

```
rule run:
    input: ["output/Luca/info.txt", "output/Fred/info.txt", “output/Guillaume/info.txt"]

rule merge_data:
    input:
        address="output/{name}/address.txt",
        phone="output/{name}/phone.txt",
    output: "output/{name}/info.txt"
    shell:"cat {input.address} > {output} && cat {input.phone} >> {output}"
```

```
rule get_info:
    input:
         exe="get_info.py",
         infile="input/{info}.txt",
    output: temp("output/Luca/{info}.txt")
    shell: "python {input.exe} {wildcards.name} {input.infile} {output}
```

Advanced tutorial: use python script

```
"""Extract information about a person."""
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("name", help="person of interest")
parser.add_argument("infile", help="text file with information")
parser.add_argument("outfile", help="where to write extracted information")
args = parser.parse_args()

# -- find information
print("extracting information...")
found = None
with open(args.infile) as f:
    for ln in f:
        if ln.startswith(args.name):
            found = ln
            break  # assume only one line with info
info = found[len(args.name):]
# -- write information
with open(args.outfile, "w") as f:
    f.write(info)
```

Task 6. Added print style information, can you figure out how to incorporate it?
Hint: Have a look how to incorporate log

Advanced tutorial: use python script

Other advanced use:

1. Often you want to run the same rule on different samples or with different options for your scripts. This can be done in snakelike using config files written in yaml
   - Define `cfg.yaml` with

```
Data:
    - 'data1.root'
    - 'data2.root'
```

Task @ home: how do you encode it in the snakefile?

2. The Snakefile can quickly grow to a monster with tens of rules. For this reason it's possible to split them into more files and then include them into the Snakefile.

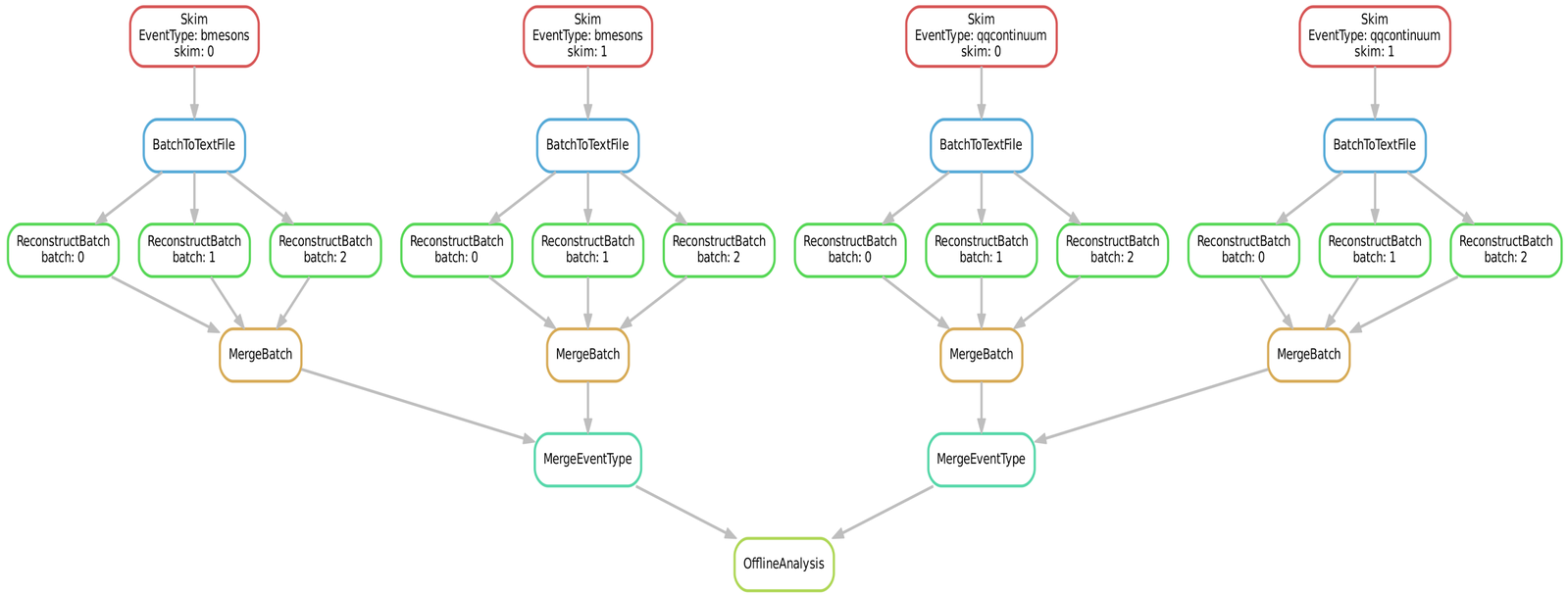
Task @ home: how do you encode it in the snakefile? Hint: use `include`

Snakemake HEP example

We want to run over data on the grid:

1. perform skim make
2. basic analysis
3. and produce histograms

Let's have an example taken from Belle II: we want to submit four skims (two for *B*-mesons and two for quark-antiquark continuum) to the grid, which are each reconstructed in three batches on the *KEKcc* batch system. The reconstructed batches are merged and finally we fill some histograms in the offline analysis.



Snakemake HEP example (Sneak peak)

```
configfile: "config.yaml"

workdir: config["outputDirectory"]

localrules: OfflineAnalysis, MergeEventType

rule OfflineAnalysis:
    input:
        data_BB = "data/projectName_bmesons/reco.root",
        data_QQ = "data/projectName_qqcontinuum/reco.root"
    output:
        mbc_plot = "Mbc.jpg",
        deltaE_plot = "deltaE.jpg"
    script:
        “offlineanalysis.py"

def PathDictionary(wildcards): #match gbasf2 dataset paths to output file paths
    Paths = dict()
    for i,skim in enumerate(open(f"{workflow.basedir}/../{wildcards.EventType}skims.dat",'r').read().splitlines()):
        Paths.update({skim:f"Reconstruction/projectName_{wildcards.EventType}/skim_{i}/reco.root"})
    return Paths

rule MergeEventType:
    input:
        unpack(PathDictionary)
    output:
        "data/projectName_{EventType}/reco.root"
    shell:
        "hadd {output} {input}" #for merging nTuples
```

Thanks for listening!

Backup

Snakemake

Instalation guidelines:

1. `conda install -n base -c conda-forge mamba` *(install a Conda-based Python3 distribution)*
2. `conda activate base`
3. `mamba create -c conda-forge -c bioconda -n snakemake` *(This will install snakemake into an isolated software environment)*
4. `conda activate snakemake` *(Activate the environment)*
5. `snakemake --help`

Short tutorial:

1. mkdir snakemake-demo
2. cd snakemake-demo
3. wget https://github.com/snakemake/snakemake-tutorial-data/archive/v5.4.5.tar.gz (Linux) / Brew install wget

5. conda activate base
6. mamba create -c conda-forge -c bioconda -n snakemake snakemake (This will install snakemake into an isolated software environment)
7. conda activate snakemake (Activate the environment)
8. snakemake --help