Developing HLT lines for Run 3 with HltEfficiencyChecker

Ross Hunter

University of Warwick, U.K

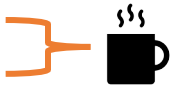
Run 3 Starterkit - 17/03/22

with many thanks to Olli Lupton, Rosen Matev, Sascha Stahl and Mika Vesterinen.

Plan

Contents

- Introduction: why you should care about rates and efficiencies & overview of HltEfficiencyChecker,
- Example 1: Rates of Marian's HLT2 lines using the "wizard",
- Example 2: Signal efficiencies of Marian's HLT2 lines with the "wizard",
- Different types of trigger efficiency,
- Example 3: Rates & efficiencies in the advanced "by-hand" workflow.



If time allows:

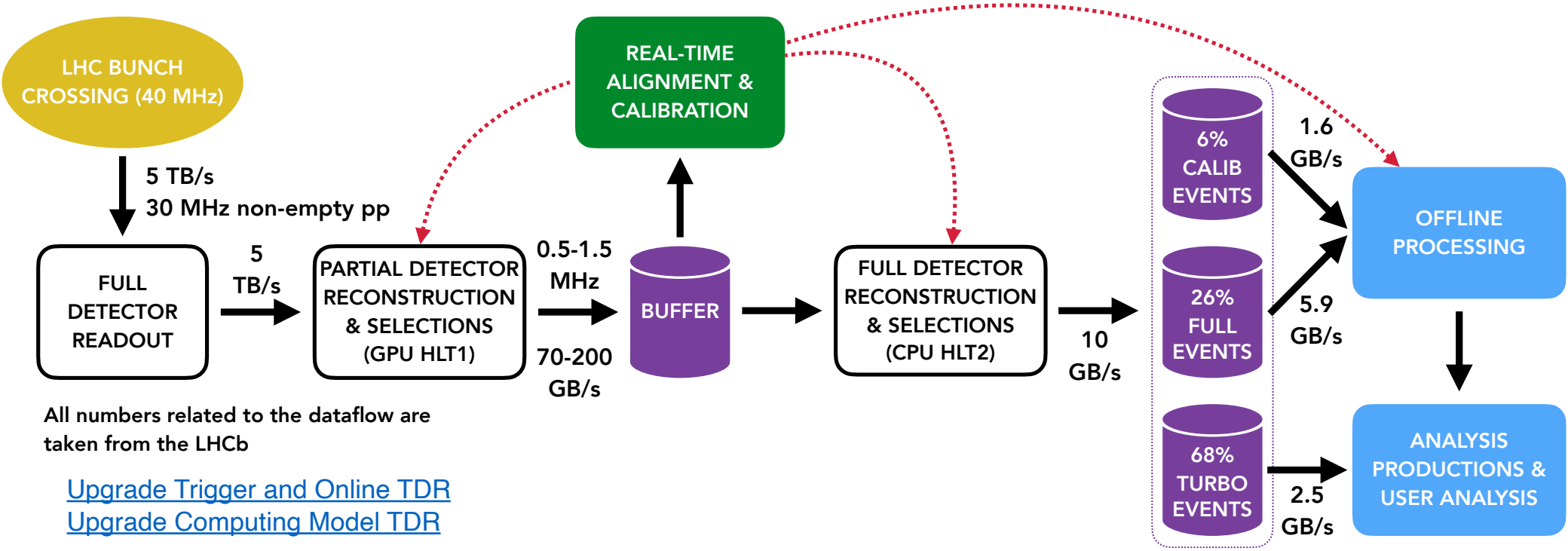
- Example 4: Running HLT1-then-HLT2 to get HLT2-given-HLT1 efficiencies,
- A few things I didn't go through,

- Summary.

- The examples will involve me coding things up live (sorry in advance for my typing). You can code along with me, or I have provided the completed examples as well. Download the slides to copy!
- Questions welcome anytime. I won't be able to see raised hands so you'll have to shout at me/someone tell me if there is a raised hand.

Intro: Why you should care about rates & efficiencies

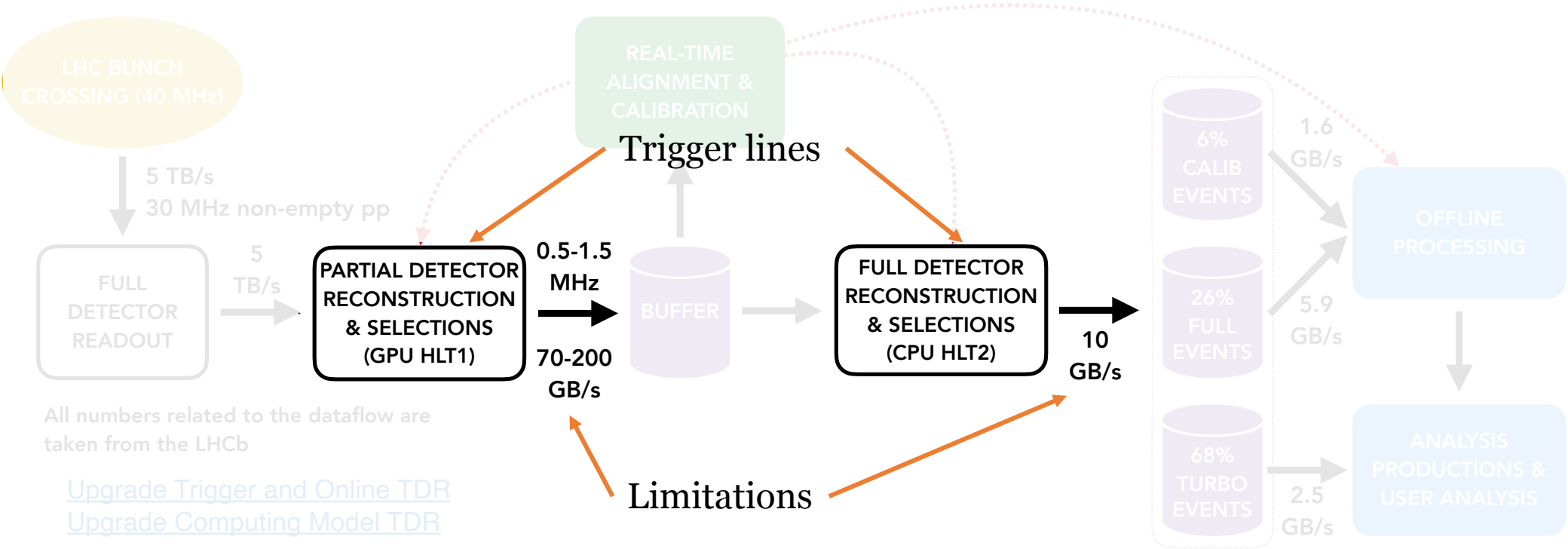
The LHCb Dataflow in Run 3



All numbers related to the dataflow are taken from the LHCb

Upgrade Trigger and Online TDR
Upgrade Computing Model TDR

The LHCb Dataflow in Run 3



Ideally, also: $\varepsilon_{interesting} = 1,\ \varepsilon_{boring} = 0$...

Overview of HltEfficiencyChecker

Wise trigger line development

(with simulated data, before taking real data)

Trigger Rate

input rate × # triggered events / # minimum bias events

(N.B Bandwidth (MB/s) = Trigger Rate (kHz) x Event Size (kB))

Signal efficiency

triggered events / # we expect to trigger on

HltEfficiencyChecker was constructed to give these metrics in a consistent, transparent, easy-to-use and automated way.

Wise trigger line development

(with simulated data, before taking real data)

Trigger Rate

input rate × # triggered events / # minimum bias events

(N.B Bandwidth (MB/s) = Trigger Rate (kHz) x Event Size (kB))

Signal efficiency

triggered events / # we expect to trigger on

Everyone uses the same definitions

HltEfficiencyChecker was constructed to give these metrics in a consistent, transparent, easy-to-use and automated way.

Thoroughly-documented, accessible & centralized code

"Beginner" and "developer" modes, tutorials etc.

Results in 1 command

Setup

- HltEfficiencyChecker lives in MooreAnalysis.

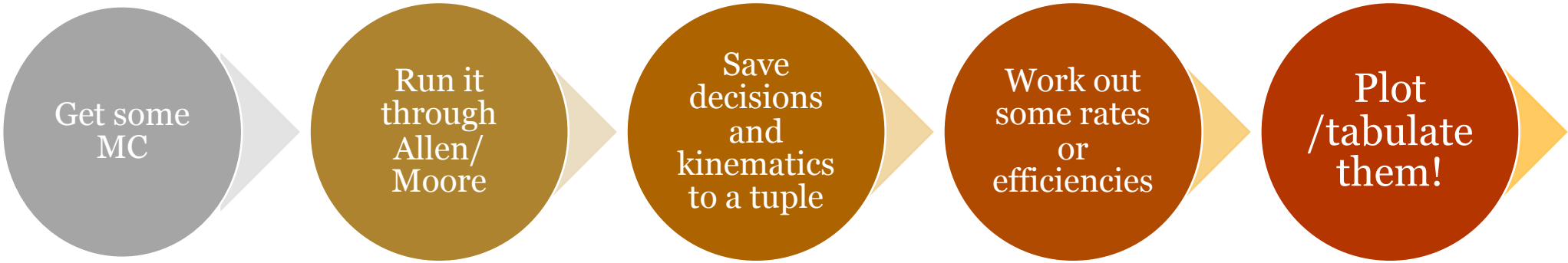
- The vanilla stack doesn't ship MooreAnalysis, so better get it. In `stack/`:
  - `make fast/MooreAnalysis`

- I'll be working from the branch rjhunter-starterkit-run-3. You can switch to the same branch with
  - `git fetch origin rjhunter-starterkit-run-3`
  - `git checkout -b rjhunter-starterkit-run-3 FETCH_HEAD`

- NOTE: File paths will always be relative to the top level of the `stack/`

HltEfficiencyChecker in principle

- Workflow:



- Configured using a single options file.
- Documentation.
- "Released" in 2020, talk.

Example 1: HLT2 “wizard” rates

- Let’s take Marian’s example further, and work out the rate and efficiency of the Starterkit $B_s \to J/\psi\phi$ lines.

- To configure HltEfficiencyChecker, we need an options file in either **.yaml** or **.py**

“Developer” mode

- Requires experience writing Moore options files,
- A simple extension to Moore options files,
- Full control/flexibility,
- A.K.A “by-hand”.

“Beginner” mode

- Designed to be as self-explanatory as possible: no experience required,
- Can’t quite do everything fancy, but will do most of what you need to tune a line,
- The “wizard” script writes a .py file from your .yaml *under-the-hood*.

Example 1: HLT2 “wizard” rates

- I’ll start from MooreAnalysis/HltEfficiencyChecker/options/hlt2_rate_example.yaml. This uses the same minimum bias MC as in Marian’s options file.

Example 1: HLT2 “wizard” rates

- I’ll start from MooreAnalysis/HltEfficiencyChecker/options/hlt2_rate_example.yaml. This uses the same minimum bias MC as in Marian’s options file.

- I’m going to (hopefully!) end up with something like MooreAnalysis/HltEfficiencyChecker/options/starterkit/hlt2_starterkit_rates.yaml

- Execute with `MooreAnalysis/run MooreAnalysis/HltEfficiencyChecker/scripts/hlt_eff_checker.py MooreAnalysis/HltEfficiencyChecker/ options/starterkit/hlt2_starterkit_rates.yaml`

- Feel free to code along with me, or pull up the completed example.

Example 1: Results

- With just 1000 events:

```
--------------------------------------------------------------------------------
INFO:    Starting /storage/epp2/phrrkw/stack/MooreAnalysis/HltEfficiencyChecker/scripts/hlt_calculate_rates.py...
--------------------------------------------------------------------------------
INFO:    No lines specified. Defaulting to all...
HLT rates:
--------------------------------------------------------------------------------
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision     Incl: 0.0  +/- 0.0  kHz, Excl: 0.0  +/- 0.0  kHz
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision  Incl: 3.0  +/- 1.72 kHz, Excl: 3.0  +/- 1.72 kHz
--------------------------------------------------------------------------------
Hlt2 Total:                                                Rate:         3 +/- 1.7 kHz
--------------------------------------------------------------------------------
Finished printing HLT rates!
```

- (prepared earlier) if instead 5000 events:

```
--------------------------------------------------------------------------------
INFO:    Starting /storage/epp2/phrrkw/stack/MooreAnalysis/HltEfficiencyChecker/scripts/hlt_calculate_rates.py...
--------------------------------------------------------------------------------
INFO:    No lines specified. Defaulting to all...
HLT rates:
--------------------------------------------------------------------------------
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision     Incl: 1.0  +/- 0.44 kHz, Excl: 0.2  +/- 0.19 kHz
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision  Incl: 1.8  +/- 0.59 kHz, Excl: 1.0  +/- 0.44 kHz
--------------------------------------------------------------------------------
Hlt2 Total:                                                Rate:         2 +/- 0.6 kHz
--------------------------------------------------------------------------------
Finished printing HLT rates!
```

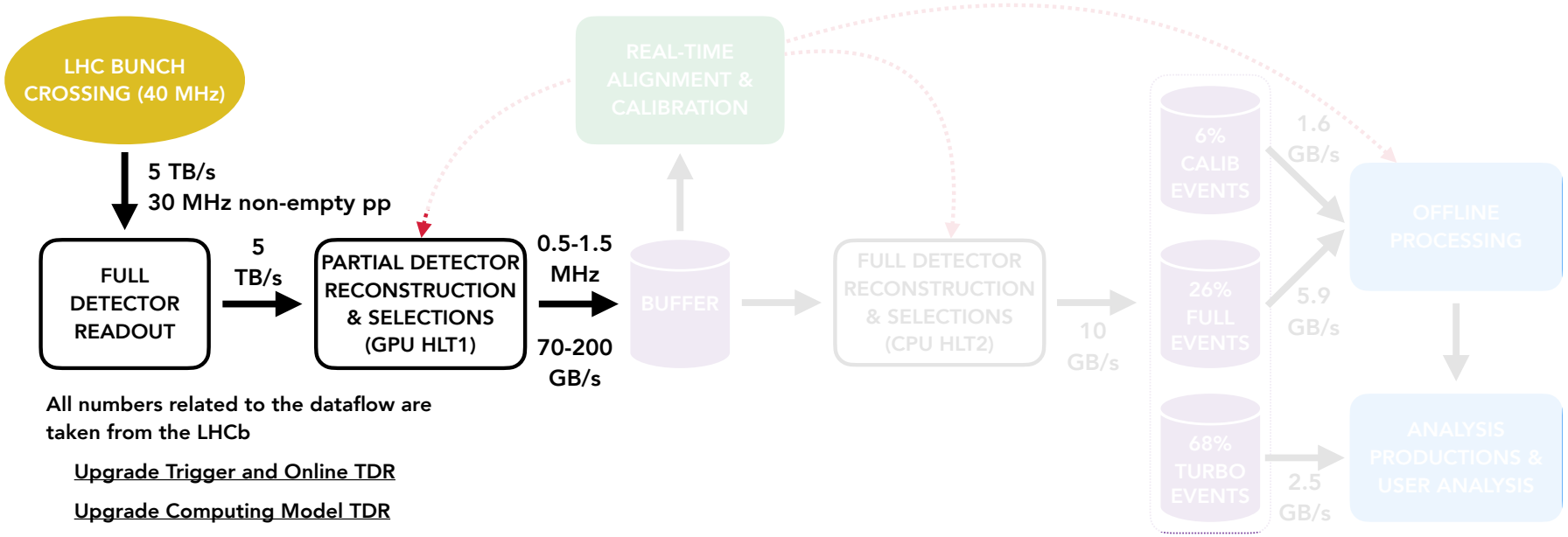
- Notice both the *inclusive* and *exclusive* rates are given…

Inclusive/exclusive and `using_hlt1_filtered_MC`

- *Inclusive* rate: rate of that line firing agnostic to other lines,
- *Exclusive* rate: rate of firing on events that only this line fired for,
- *Total* rate: rate that one or more lines fired on an event

These give you some sense of the overlap between lines.

- `using_hlt1_filtered_MC` - a roughly representative HLT1 has already filtered the file, so the input rate is assumed to be 1 MHz (expected HLT1 output rate in Run 3).



“Wizard” Trigger efficiencies

“Wizard” Trigger efficiencies

- Need another .yaml, with a few key differences. Firstly, need signal MC and a MC decay descriptor (1).

“Wizard” Trigger efficiencies

- Need another .yaml, with a few key differences. Firstly, need signal MC and a MC decay descriptor (1).

- We also must consider denominators.
  Signal efficiency = # triggered events / # we expect to trigger on

Subjective!

“Wizard” Trigger efficiencies

- Need another .yaml, with a few key differences. Firstly, need signal MC and a MC decay descriptor (1).

- We also must consider denominators.

Subjective!

Signal efficiency = # triggered events / # we expect to trigger on

- HltEfficiencyChecker provides a few “standard” denominators. The default is called CanRecoChildren (C.R.C):
  - Events where all final-state particles are charged & made long tracks in $2 < \eta < 5$,
  - You have to tell HltEfficiencyChecker what these are called! (2)

“Wizard” Trigger efficiencies

- Need another .yaml, with a few key differences. Firstly, need signal MC and a MC decay descriptor (1).

- We also must consider denominators. Subjective!
  Signal efficiency = # triggered events / # we expect to trigger on

- HltEfficiencyChecker provides a few “standard” denominators. The default is called CanRecoChildren (C.R.C):
  - Events where all final-state particles are charged & made long tracks in $2 < \eta < 5$,
  - You have to tell HltEfficiencyChecker what these are called! (2)

- (1) and (2) are specified together in an annotated decay descriptor e.g.

```
"${Bs}[B_s0 => ( J/psi(1S) => ${mup}mu+ ${mum}mu- ) ( phi(1020) => ${Kp}K+ ${Km}K- )]CC”
```

and the `reconstructible_children` key e.g. `mup,mum,Kp,Km`

Example 2: HLT2 “wizard” efficiencies

- Let’s code up our first options file! I’ll start from MooreAnalysis/HltEfficiencyChecker/options/hlt2_eff_example.yaml, and use Marian’s MC files that he put in Moore/Hlt/Hlt2Conf/options/run_starterkit_bs_to_jpsiphi.py

- I’m going to end up with something like MooreAnalysis/HltEfficiencyChecker/options/starterkit/hlt2_starterkit_effs.yaml

- Feel free to code along with me, or pull up the completed example.

- Execute with e.g.

```
MooreAnalysis/run MooreAnalysis/HltEfficiencyChecker/scripts/hlt_eff_checker.py
MooreAnalysis/HltEfficiencyChecker/options/starterkit/hlt2_starterkit_effs.yaml
```

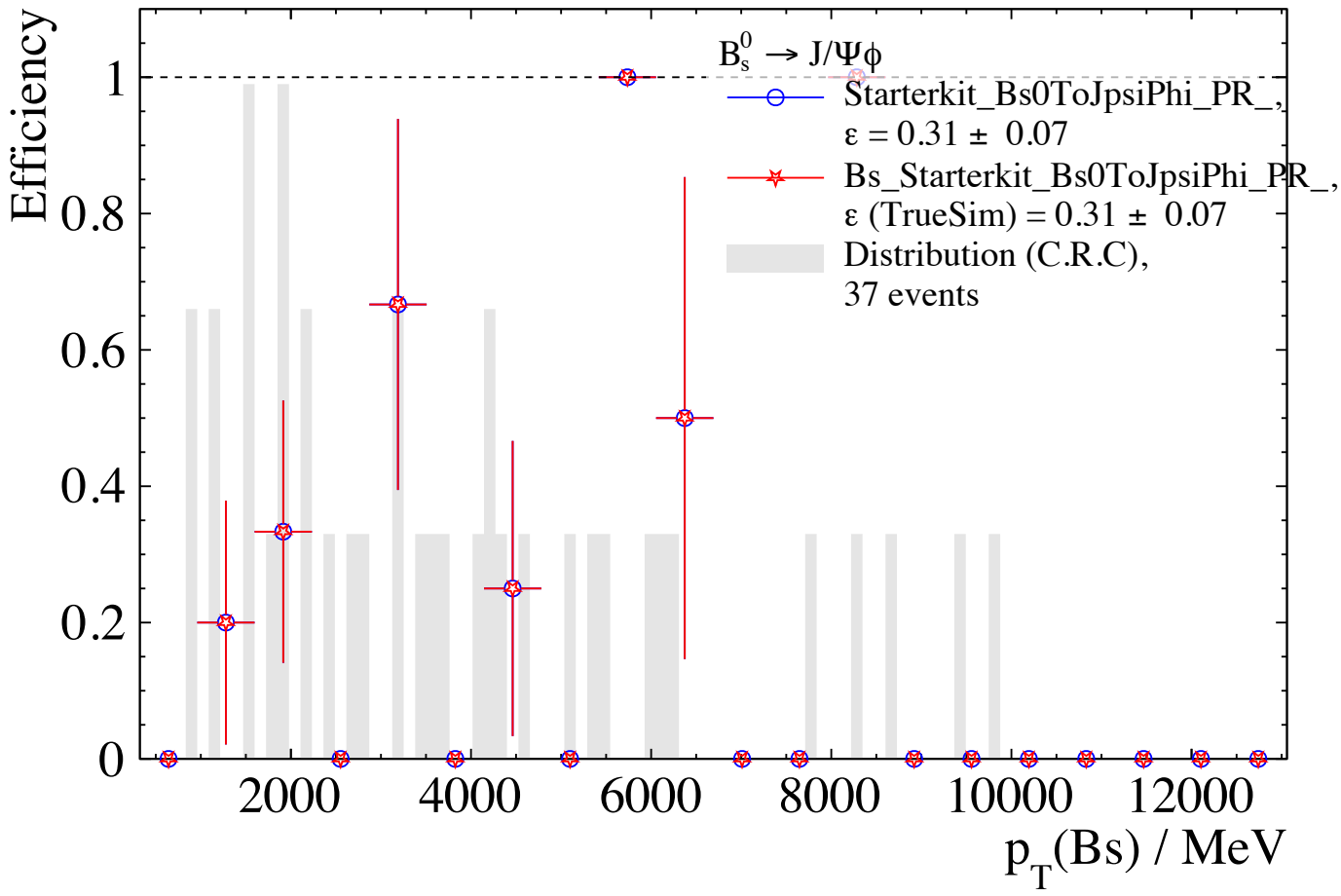
Live Example 2: Results

- With 100 events:

```
--------------------------------------------------------------------------------
INFO:    Starting /storage/epp2/phrrkw/stack/MooreAnalysis/HltEfficiencyChecker/scripts/hlt_line_efficiencies.py...
--------------------------------------------------------------------------------
INFO:    No lines specified. Defaulting to all...
Integrated HLT efficiencies for the lines with denominator: CanRecoChildren
--------------------------------------------------------------------------------
Line:    Bs_Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecisionTrueSim          Efficiency:     0.308 +/- 0.074
Line:    Bs_Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecisionTrueSim       Efficiency:     0.564 +/- 0.079
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision                    Efficiency:     0.308 +/- 0.074
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision                 Efficiency:     0.564 +/- 0.079
--------------------------------------------------------------------------------
Finished printing integrated HLT efficiencies for denominator: CanRecoChildren
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
INFO:    Making plots...
--------------------------------------------------------------------------------
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
INFO:    --xtitle not specified. Using branch name and attempting to figure out the units...
INFO:    Found unit "MeV" for variable "PT".
INFO:    Now plotting...
Info in <TCanvas::Print>: pdf file Efficiencies__Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision__CanRecoChildren__logPT.pdf has been created
Info in <TCanvas::Print>: pdf file Efficiencies__Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision__CanRecoChildren__PT.pdf has been created
Info in <TCanvas::Print>: pdf file Efficiencies__Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision__CanRecoChildren__logPT.pdf has been created
Info in <TCanvas::Print>: pdf file Efficiencies__Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision__CanRecoChildren__PT.pdf has been created
Info in <TCanvas::Print>: pdf file Bs__TrueSimEfficiencies__CanRecoChildren__AllLines__logPT.pdf has been created
Info in <TCanvas::Print>: pdf file Bs__TrueSimEfficiencies__CanRecoChildren__AllLines__PT.pdf has been created
--------------------------------------------------------------------------------
INFO     Finished making plots. Goodbye.
--------------------------------------------------------------------------------
```

Live Example 2: Results

- With 100 events: opening
  Efficiencies__Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision__CanRecoChildren__PT.pdf



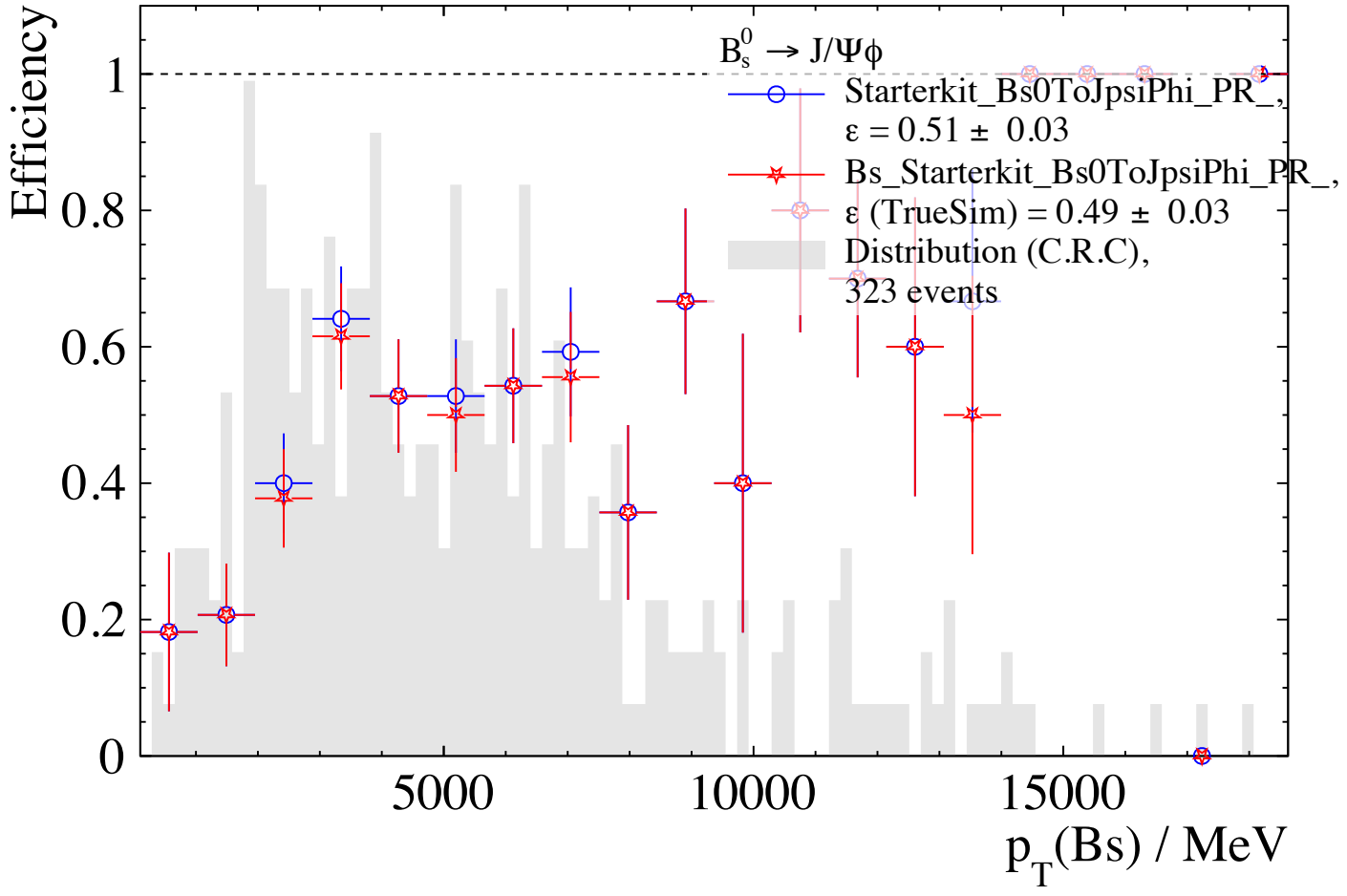
Live Example 2: Results

- With 1000 events:

```
--------------------------------------------------------------------------------
INFO:    Starting /storage/epp2/phrrkw/stack/MooreAnalysis/HltEfficiencyChecker/scripts/hlt_line_efficiencies.py...
--------------------------------------------------------------------------------
INFO:    No lines specified. Defaulting to all...
Integrated HLT efficiencies for the lines with denominator: CanRecoChildren
--------------------------------------------------------------------------------
Line:    Bs_Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecisionTrueSim          Efficiency:     0.494 +/- 0.028
Line:    Bs_Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecisionTrueSim       Efficiency:     0.606 +/- 0.027
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision                    Efficiency:     0.509 +/- 0.028
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision                 Efficiency:     0.621 +/- 0.027
--------------------------------------------------------------------------------
Finished printing integrated HLT efficiencies for denominator: CanRecoChildren
--------------------------------------------------------------------------------
```

Live Example 2: Results

- With 1000 events:



Before we take a break…

- Later on I'll show some features that are best displayed with a few thousand events, which takes a few minutes or so using the full HLT2 reconstruction.

- I ran these and placed them in the checker-skit-rates/ and checker-skit-effs/ directories in /afs/cern.ch/user/r/rjhunter/public. If you're coding along with me, please copy these directories for use later.
  - On lxplus: `cp -r /afs/cern.ch/user/r/rjhunter/public/checker-skit-* .`

The Wizardry and --dry-run

- The --dry-run argument is your friend!

```
[phrrkw@veleta stack]$ MooreAnalysis/run MooreAnalysis/HltEfficiencyChecker/scripts/hlt_eff_checker.py MooreAnalysis/HltEfficiencyChecker/options/starterkit/h
lt2_starterkit_rates.yaml --dry-run
The commands to run are...
cd 'checker-20220314-174152'
'/storage/epp2/phrrkw/stack/MooreAnalysis/run' 'gaudirun.py' '.hlt_eff_checker_options_D3A813F4.py'
'/storage/epp2/phrrkw/stack/MooreAnalysis/run' '/storage/epp2/phrrkw/stack/MooreAnalysis/HltEfficiencyChecker/scripts/hlt_calculate_rates.py' 'hlt2_starterkit
_rate_ntuple.root' '--using-hlt1-filtered-MC' '--json=Hlt2_rates.json'
```

1) It shows you what goes on under-the-hood
2) If you just want to re-run the analysis (i.e. make some different plots) then you only need to re-run the final (fast) step, not the tuple-making (slow) step. Save time!

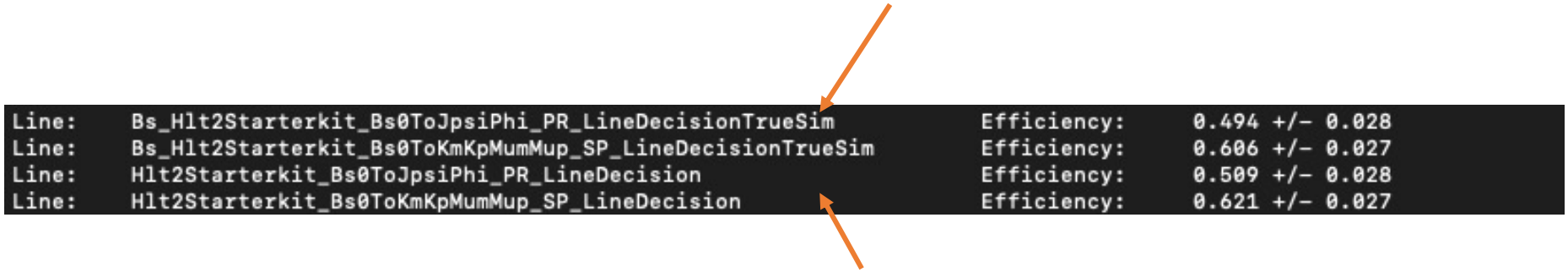
Temporary directories

- Results/plots will be dumped into a temporary directory e.g. checker-20220315-111521/. You can change the name of this with args to hlt_eff_checker.py

```
[phrrkw@veleta stack]$ MooreAnalysis/run MooreAnalysis/HltEfficiencyChecker/scripts/hlt_eff_checker.py -h
usage: hlt_eff_checker.py [-h] [-o OUTPUT] [-s OUTPUT_SUFFIX] [--force] [-n] [--ignore-broken-inputs] config

Script that extracts and plots efficiencies and rates from the high level trigger.

positional arguments:
  config                YAML configuration file

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        Output directory prefix. See also --output-suffix.
  -s OUTPUT_SUFFIX, --output-suffix OUTPUT_SUFFIX
                        Output directory suffix
  --force               Do not fail when output directory exists
  -n, --dry-run         Only print the commands needed to run from stack/ directory.
  --ignore-broken-inputs
                        Ignore Gaudi::ReturnCode::FailInput errors in check_call. Tupling already skips these files, so this option allows the rest of HltEfficiencyChecker to proceed after skipping a
                        broken input file.
```

- e.g. `-o checker-skit- -s rates` will give checker-skit-rates/

“TrueSim” efficiencies



```
Line:    Bs_Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecisionTrueSim          Efficiency:    0.494 +/- 0.028
Line:    Bs_Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecisionTrueSim       Efficiency:    0.606 +/- 0.027
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision                    Efficiency:    0.509 +/- 0.028
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision                 Efficiency:    0.621 +/- 0.027
```

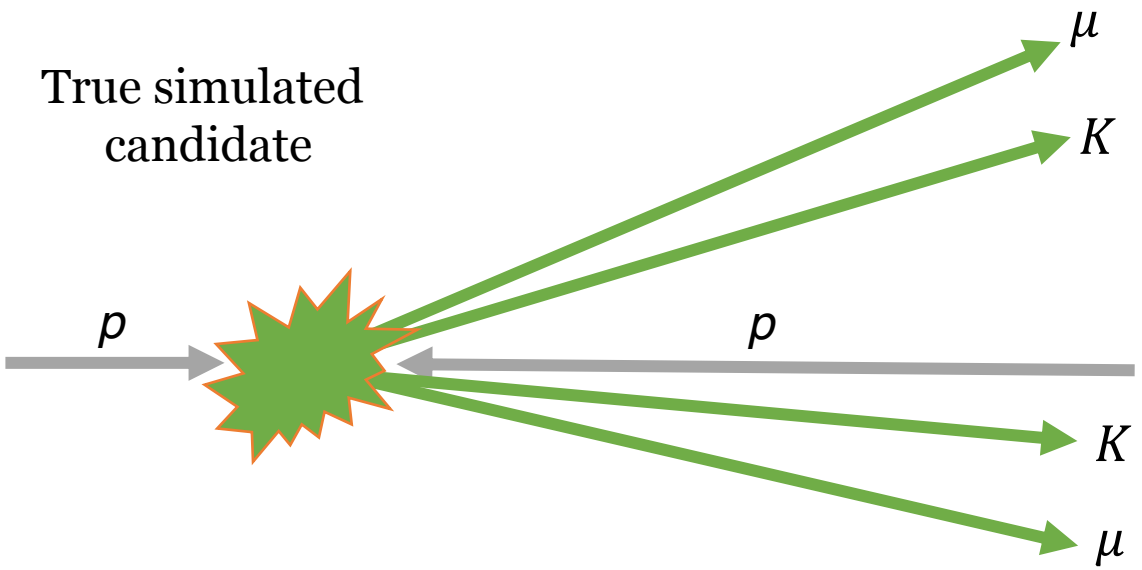
The problem with efficiencies

- The signal “decision” (DEC)* efficiency does not give the full picture.

* # triggered events /
# we expect to trigger on

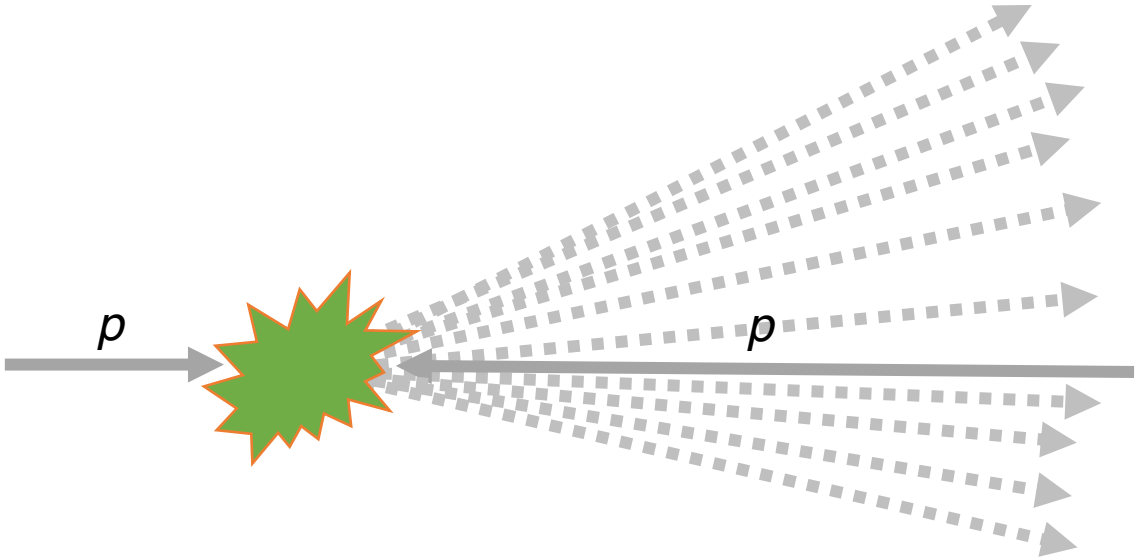
The problem with efficiencies

- The signal "decision" (DEC) efficiency does not give the full picture.
- Consider $B_s \to J/\Psi\,\phi$ with $(\phi \to K^+K^-)$ and $(J/\Psi \to \mu^+\mu^-)$...



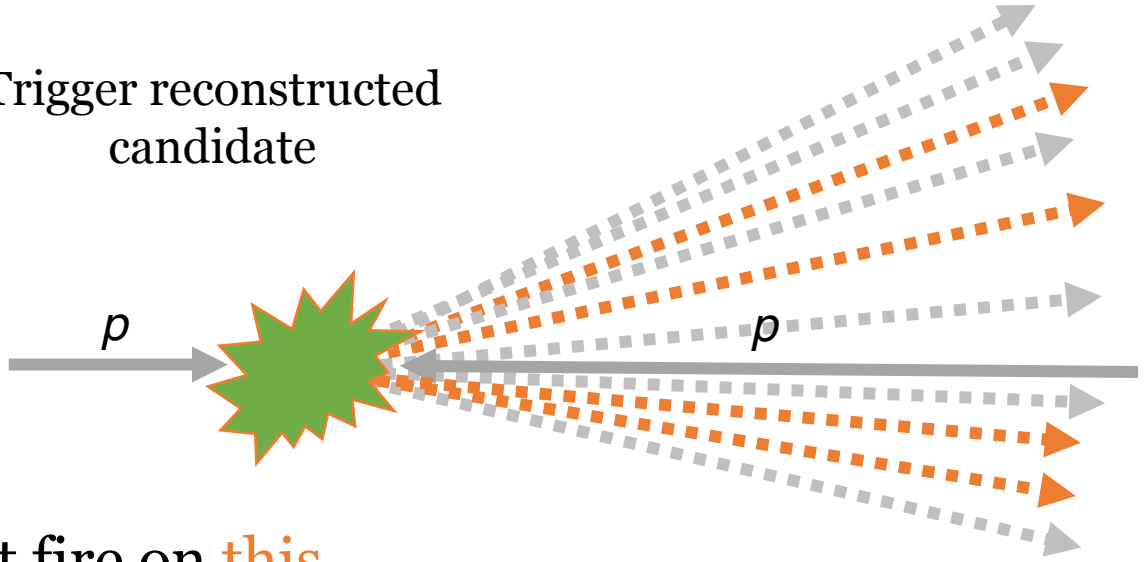
The problem with efficiencies

- The signal "decision" (DEC) efficiency does not give the full picture…
- Consider $B_s \to J/\Psi\, \phi$ with $(\phi \to K^+K^-)$ and $(J/\Psi \to \mu^+\mu^-)$…
- … in a hadron collider:



The problem with efficiencies

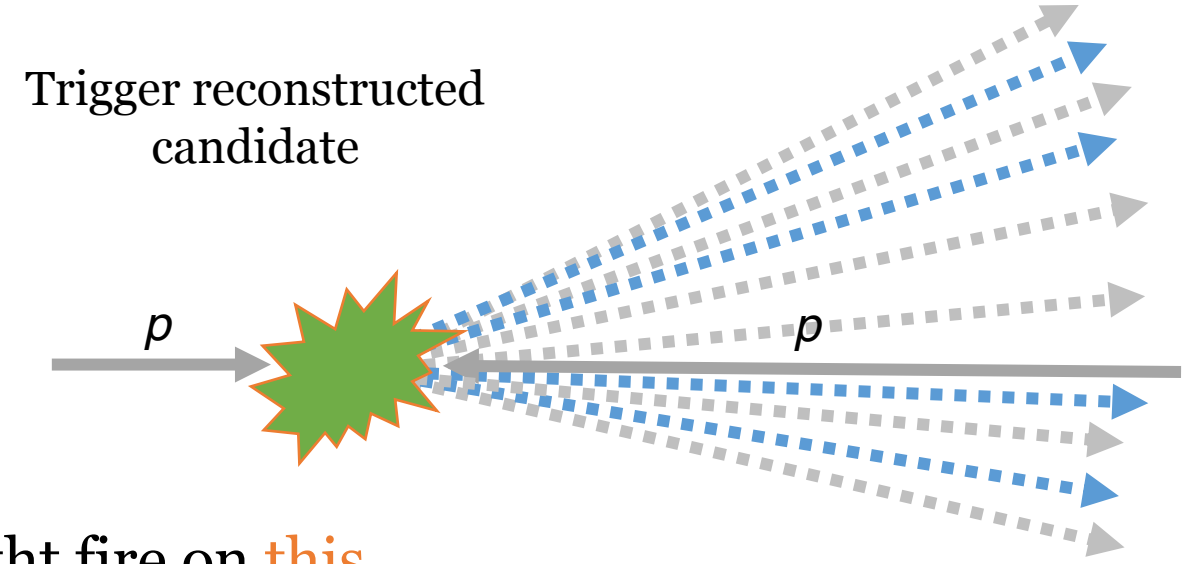
- The signal "decision" (DEC) efficiency does not give the full picture…
- Consider $B_s \to J/\Psi\,\phi$ with $(\phi \to K^+K^-)$ and $(J/\Psi \to \mu^+\mu^-)$…
- … in a hadron collider:



- Trigger might fire on this,

The problem with efficiencies

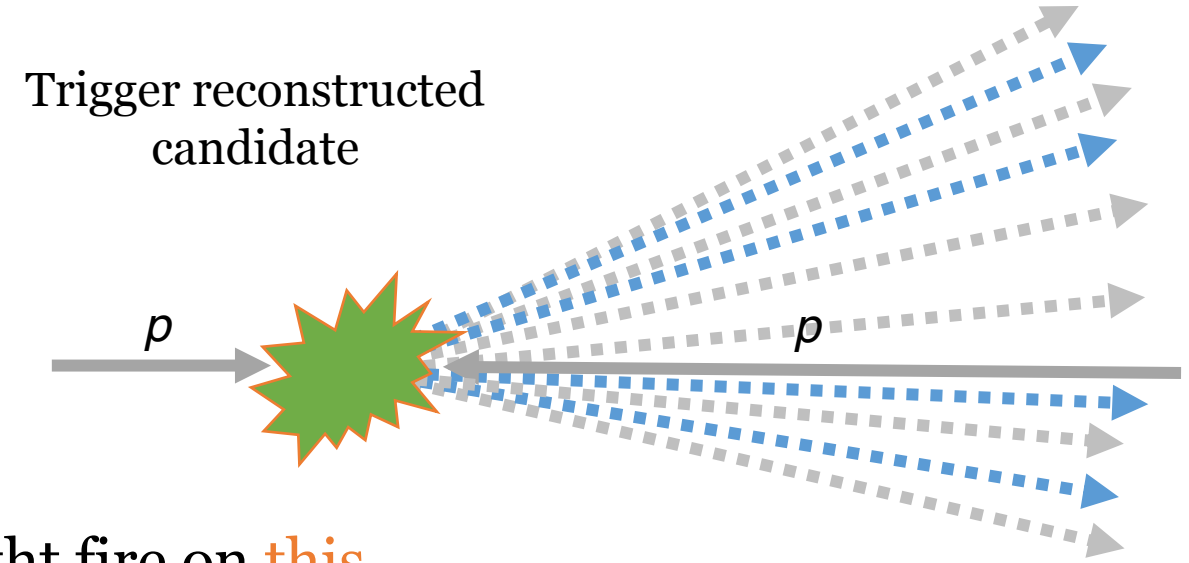
- The signal "decision" (DEC) efficiency does not give the full picture…
- Consider $B_s \to J/\Psi\,\phi$ with $(\phi \to K^+K^-)$ and $(J/\Psi \to \mu^+\mu^-)$…
- … in a hadron collider:



- Trigger might fire on this,
- Or this…

The problem with efficiencies

- The signal "decision" (DEC) efficiency does not give the full picture…
- Consider $B_s \to J/\Psi\,\phi$ with $(\phi \to K^+K^-)$ and $(J/\Psi \to \mu^+\mu^-)$…
- … in a hadron collider:



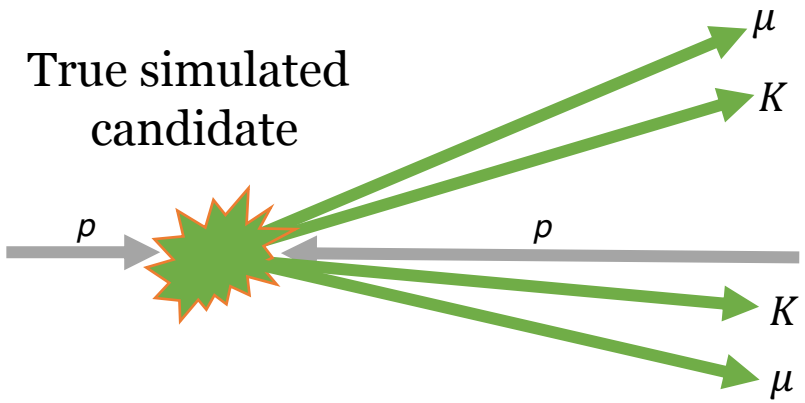
- Trigger might fire on this,
- Or this…

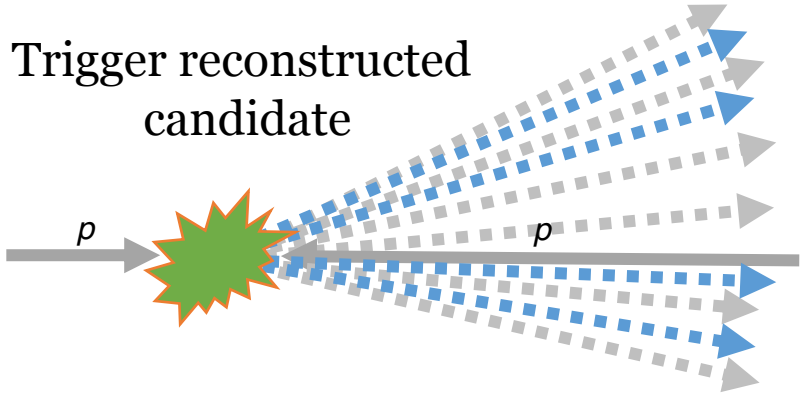
✗ Both of these count towards the DEC efficiency, signal or not.

✓ Need an efficiency that counts **only signal triggers**.

A better efficiency

TrueSim eff. = # triggered & **matched** events / # we expect to trigger on






```
simulated_hits =
    {kaon1: [hit1, hit2, ...],
     ...
     muon2: [hit12, hit13, ...]}
```

```
reconstructed_hits =
    {track1: [hit1', hit2', ...],
     ...
     track4: [hit12', hit13', ...]}
```

Track is matched if #overlapping hits ≥ 70% of trigger candidate hits

If all tracks match: trigger fired on the signal particle in that event.

Choice motivated by study presented at vCHEP.

The “by-hand” workflow

The “by-hand” workflow

- Let’s take Marian’s example further, and work out the rate and efficiency of the Starterkit $B_s \to J/\psi\phi$ lines.

- To configure HltEfficiencyChecker, we need an options file in either **.yaml** or **.py**

“Developer” mode

- Requires experience writing Moore options files,
- A simple extension to Moore options files,
- Full control/flexibility,
- A.K.A “by-hand”.

“Beginner” mode

- Designed to be as self-explanatory as possible: no experience required,
- Can’t quite do everything fancy, but will do most of what you need to tune a line,
- Writes a .py file from your .yaml *under-the-hood*,
- A.K.A the “wizard”.

Example 3: efficiencies by-hand

- To show that this really is a small extension to a Moore options file, let's hack Marian's options file (Moore/Hlt/Hlt2Conf/options/run_starterkit_bs_to_jpsiphi.py) to give us a tuple that we can calculate rates and efficiencies from.
  - Make sure to set `minbias=False` for efficiencies.

- I'll end up with something like MooreAnalysis/HltEfficiencyChecker/options/starterkit/hlt2_starterkit_example.py
  - Feel free to code along or just look at this example!

- Run with

```
MooreAnalysis/run gaudirun.py
MooreAnalysis/HltEfficiencyChecker/options/starterkit/hlt2_starterkit_example.py
```

Example 3: efficiencies by-hand

- The output of the tupling job should look something like:

```
*.....................................................................*
*Br  154 :Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision :               *
*         | Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision/I            *
*Entries :      100 : Total  Size=       1162 bytes  File Size  =        184 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   2.81     *
*.....................................................................*
*Br  155 :Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision :            *
*         | Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision/I         *
*Entries :      100 : Total  Size=       1177 bytes  File Size  =        194 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   2.68     *
*.....................................................................*
*Br  156 :Hlt2nSelections : Hlt2nSelections/i                         *
*Entries :      100 : Total  Size=       1022 bytes  File Size  =        170 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   2.88     *
*.....................................................................*
NTupleSvc                                          INFO NTuples saved successfully
ApplicationMgr                                     INFO Application Manager Finalized successfully
ApplicationMgr                                     INFO Application Manager Terminated successfully
```

- This tuple has trigger decisions (+MC truth information if `decay` is specified). A further step is needed to get your rates/efficiencies…

Demystifying the spells

- The wizard script (hlt_eff_checker.py) calls the previous command, and then the analysis script. “By-hand”, we must call the analysis python script ourselves:

```
$ MooreAnalysis/run python MooreAnalysis/HltEfficiencyChecker/scripts/hlt_line_efficiencies.py <args>
$ MooreAnalysis/run python MooreAnalysis/HltEfficiencyChecker/scripts/hlt_calculate_rates.py <args>
```

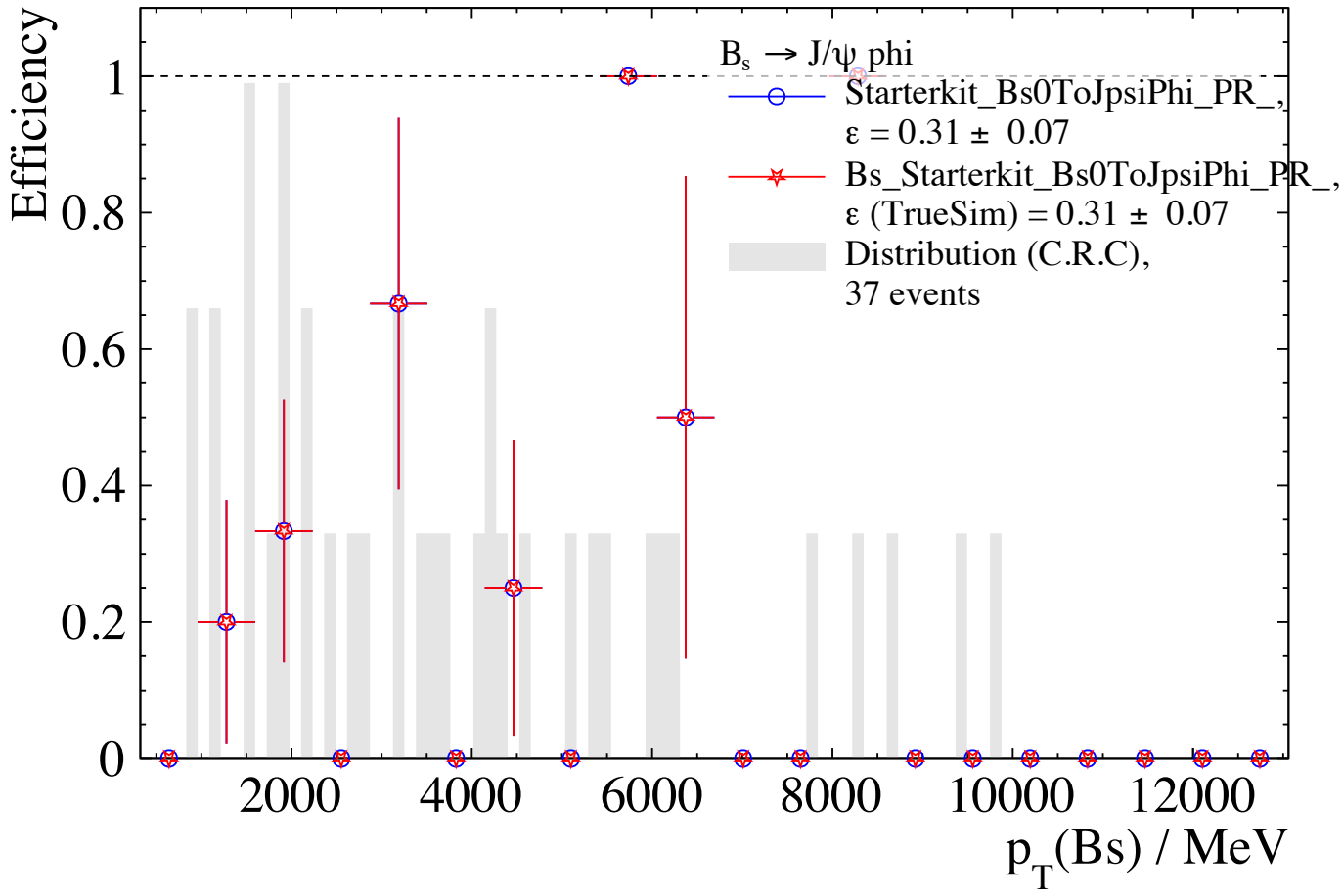
- `<args>`? The analysis key-value pairs in the yaml correspond to command-line arguments to the rate or efficiency-calculating scripts:

```
ntuple_path: &NTUPLE hlt2_starterkit_eff_ntuple.root
analysis:
    script: $HLTEFFICIENCYCHECKERROOT/scripts/hlt_line_efficiencies.py
    args:
        input: *NTUPLE
        reconstructible_children: mup,mum,Kp,Km
        # The parent is automatically deduced from the annotated decay descriptor,
        # which is passed in a file such as eff_ntuple.root.json, but can also be given:
        # parent: B_s0
        legend_header: "B^{0}_{s} #rightarrow J/#Psi#phi"
        true_signal_to_match_to: "Bs"
        make_plots: true
```

- `MooreAnalysis/run python MooreAnalysis/HltEfficiencyChecker/scripts/hlt_line_efficiencies.py hlt2_starterkit_eff_ntuple.root --reconstructible-children mup,mum,Kp,Km --legend-header “B^{0}_{s} #rightarrow J/#psi #phi” --true-signal-to-match-to Bs --make-plots`

Example 3: Results

- It's the same as before! Who knew?!?!



It’s all the same under-the-hood…

- We just did

```
MooreAnalysis/run gaudirun.py
MooreAnalysis/HltEfficiencyChecker/options/starterkit/hlt2_starterkit_example.py
```

- Followed by

```
MooreAnalysis/run python
MooreAnalysis/HltEfficiencyChecker/scripts/hlt_line_efficiencies.py
hlt2_starterkit_eff_ntuple.root --reconstructible-children mup,mum,Kp,Km
--legend-header “B^{0}_{s} #rightarrow J/#psi #phi” --true-signal-to-match-to Bs
--make-plots
```

- C.f. what we got with `--dry-run` on the wizard example:

```
[phrrkw@veleta stack]$ MooreAnalysis/run MooreAnalysis/HltEfficiencyChecker/scripts/hlt_eff_checker.py  MooreAnalysis/HltEfficiencyChecker/options/starterkit/hlt2_starterkit_effs
.yaml --dry-run
The commands to run are...
cd 'checker-20220315-165245'
'/storage/epp2/phrrkw/stack/MooreAnalysis/run' 'gaudirun.py' '.hlt_eff_checker_options_B57EB5D1.py'
'/storage/epp2/phrrkw/stack/MooreAnalysis/run' '/storage/epp2/phrrkw/stack/MooreAnalysis/HltEfficiencyChecker/scripts/hlt_line_efficiencies.py' 'hlt2_starterkit_eff_ntuple.root'
'--reconstructible-children=mup,mum,Kp,Km' '--true-signal-to-match-to=Bs' '--make-plots' '--legend-header=B^{0}_{s} #rightarrow J/#Psi#phi'
```

Example 4: HLT1-then-HLT2

Many thanks to Matt Kenzie and Matt Monk
for their help developing this.

Chained HLT1 + HLT2

- On real data, HLT2 will be evaluated on output of HLT1. So, to tune a HLT2 line, you might want the HLT2-given-passed-HLT1 efficiency, right?

- HltEfficiencyChecker can do this too with a few extra lines, by running HLT1 and HLT2 in sequence.

- Let's take our HLT2 wizard efficiencies script (MooreAnalysis/HltEfficiencyChecker/options/starterkit/hlt2_starterkit_effs.yaml), and an example from the stack (MooreAnalysis/HltEfficiencyChecker/options/hlt1_and_hlt2_eff_example.yaml) and make the necessary additions.

- I'll end up with something like HltEfficiencyChecker/options/starterkit/hlt1_and_hlt2_starterkit_effs.yaml

Example 4: Results

- With 100 events:

```
--------------------------------------------------------------------------------
Line:    Bs_Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecisionTrueSim          Efficiency:     0.308 +/- 0.074
Line:    Bs_Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecisionTrueSim       Efficiency:     0.564 +/- 0.079
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision                    Efficiency:     0.308 +/- 0.074
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision                 Efficiency:     0.564 +/- 0.079
--------------------------------------------------------------------------------
Finished printing integrated HLT efficiencies for denominator: CanRecoChildren
--------------------------------------------------------------------------------
Integrated HLT efficiencies for the lines with denominator: CanRecoChildrenAndHlt1trackmvas
--------------------------------------------------------------------------------
Line:    Bs_Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecisionTrueSim          Efficiency:     0.292 +/- 0.093
Line:    Bs_Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecisionTrueSim       Efficiency:     0.500 +/- 0.102
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision                    Efficiency:     0.292 +/- 0.093
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision                 Efficiency:     0.500 +/- 0.102
--------------------------------------------------------------------------------
Finished printing integrated HLT efficiencies for denominator: CanRecoChildrenAndHlt1trackmvas
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
INFO:    Making plots...
--------------------------------------------------------------------------------
```

Example 4: Results

- With 1000 events:

```
--------------------------------------------------------------------------------
Line:    Bs_Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecisionTrueSim          Efficiency:     0.494 +/- 0.028
Line:    Bs_Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecisionTrueSim       Efficiency:     0.606 +/- 0.027
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision                    Efficiency:     0.509 +/- 0.028
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision                 Efficiency:     0.621 +/- 0.027
--------------------------------------------------------------------------------
Finished printing integrated HLT efficiencies for denominator: CanRecoChildren
--------------------------------------------------------------------------------
Integrated HLT efficiencies for the lines with denominator: CanRecoChildrenAndHlt1trackmvas
--------------------------------------------------------------------------------
Line:    Bs_Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecisionTrueSim          Efficiency:     0.525 +/- 0.035
Line:    Bs_Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecisionTrueSim       Efficiency:     0.625 +/- 0.034
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision                    Efficiency:     0.540 +/- 0.035
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision                 Efficiency:     0.645 +/- 0.034
--------------------------------------------------------------------------------
Finished printing integrated HLT efficiencies for denominator: CanRecoChildrenAndHlt1trackmvas
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
INFO:    Making plots...
--------------------------------------------------------------------------------
```

- Notice that the efficiencies are higher given a HLT1 MVA line passed.

Rate & efficiency of a group of lines

- If you think your signal might trigger several lines and want the inclusive rate/efficiencies, we use the handy `--rates-groups` and `--effs-groups` (thanks @lpica) arguments to the analysis scripts.

- We don't need to remake any tuples. Let's use those results directories you copied earlier! Firstly checker-skit-rates/ and translate the yaml key-value pairs to arguments to hlt_calculate_rates.py

- Let's calculate the inclusive rate of our two trigger lines using, adding the argument:

```
--rates-groups StarterkitLines:Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision,Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision
```

Rate & efficiency of a group of lines

- With 5000 events:

```
HLT rates:
-----------------------------------------------------------------------------------------------------
Line:    Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision     Incl: 1.0  +/- 0.44 kHz, Excl: 0.2  +/- 0.19 kHz
Line:    Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision  Incl: 1.8  +/- 0.59 kHz, Excl: 1.0  +/- 0.44 kHz
-----------------------------------------------------------------------------------------------------
Rates grouped by common physics:
INFO:    StarterkitLines group is formed of Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecision, Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecision.
-----------------------------------------------------------------------------------------------------
Group:   StarterkitLines lines                           Incl: 2.0  +/- 0.63 kHz, Excl: 2.0  +/- 0.63 kHz
-----------------------------------------------------------------------------------------------------
Hlt2 Total:                                               Rate:         2 +/- 0.6 kHz
-----------------------------------------------------------------------------------------------------
```

Efficiency of a group of lines

- Again, we can use the HLT1-and-HLT2 tuple we created earlier made with hlt1_and_hlt2_starterkit_effs.yaml, saved in checker-skit-effs/

Efficiency of a group of lines

- Again, we can use the HLT1-and-HLT2 tuple we created earlier made with hlt1_and_hlt2_starterkit_effs.yaml, saved in checker-skit-effs/

- Passing in addition `--effs-groups Hlt1TrackMVAs:Hlt1TwoTrackMVADecision,Hlt1TrackMVADecision` to hlt_line_efficiencies.py:

```
--------------------------------------------------------------------------------
Line:    Bs_Hlt1TrackMVADecisionTrueSim          Efficiency:     0.324 +/- 0.026
Line:    Bs_Hlt1TwoTrackMVADecisionTrueSim       Efficiency:     0.555 +/- 0.027
Line:    Hlt1TrackMVADecision                    Efficiency:     0.352 +/- 0.026
Line:    Hlt1TwoTrackMVADecision                 Efficiency:     0.597 +/- 0.027
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Efficiencies grouped by common physics:
INFO:    Hlt1TrackMVA group contains ['Hlt1TwoTrackMVADecision', 'Hlt1TrackMVADecision'].
INFO:    Hlt1TrackMVATrueSim group contains ['Bs_Hlt1TwoTrackMVADecisionTrueSim', 'Bs_Hlt1TrackMVADecisionTrueSim'].
--------------------------------------------------------------------------------
Group:   Hlt1TrackMVA lines                      Efficiency:     0.606 +/- 0.027
Group:   Hlt1TrackMVATrueSim lines               Efficiency:     0.576 +/- 0.027
--------------------------------------------------------------------------------
```

Rate & efficiency of a group of lines

- Now we have all the ingredients for a total HLT efficiency!

- Our HLT2-given-HLT1 efficiency earlier was

```
Integrated HLT efficiencies for the lines with denominator: CanRecoChildrenAndHlt1trackmvas
--------------------------------------------------------------------------------
Line:    Bs_Hlt2Starterkit_Bs0ToJpsiPhi_PR_LineDecisionTrueSim          Efficiency:     0.525 +/- 0.035
Line:    Bs_Hlt2Starterkit_Bs0ToKmKpMumMup_SP_LineDecisionTrueSim       Efficiency:     0.625 +/- 0.034
```

- The HLT1TrackMVA efficiency was

```
INFO:    Hlt1TrackMVATrueSim group contains ['Bs_Hlt1TwoTrackMVADecisionTrueSim', 'Bs_Hlt1TrackMVADecisionTrueSim'].
--------------------------------------------------------------------------------
Group:   Hlt1TrackMVA lines                     Efficiency:     0.606 +/- 0.027
Group:   Hlt1TrackMVATrueSim lines              Efficiency:     0.576 +/- 0.027
--------------------------------------------------------------------------------
```

- **Total HLT trigger efficiency** = 0.525 (0.625) x 0.576 = 0.302 (0.36)

Extra features to explore

Extra features to explore

1. Have a look through the arguments to hlt_calculate_rates.py and hlt_line_efficiencies.py e.g.

```
[phrrkw@veleta stack]$ MooreAnalysis/run python MooreAnalysis/HltEfficiencyChecker/scripts/hlt_line_efficiencies.py -h
usage: hlt_line_efficiencies.py [-h] [--metadata METADATA] [--denoms DENOMS] [--lines LINES] [--vars VARS] [--decay-name DECAY_NAME] [--parent PARENT] --reconstructible-children RECONSTRUCTIBLE_CHILDREN
                                [--true-signal-to-match-to TRUE_SIGNAL_TO_MATCH_TO] [--effs-groups EFFS_GROUPS [EFFS_GROUPS ...]] [--plot-group-effs] [--custom-denoms CUSTOM_DENOMS] [--make-plots]
                                [--legend-header LEGEND_HEADER] [--xtitle XTITLE] [--plot-format PLOT_FORMAT] [--plot-prefix PLOT_PREFIX] [--shape-distr-with-denom SHAPE_DISTR_WITH_DENOM]
                                [--nbins NBINS] [--shape-nbins SHAPE_NBINS] [--no-legend] [--json JSON]
                                input

Analysis script to calculate and plot efficiencies from the tuple provided. Use --help for options or visit the Moore online documentation.

positional arguments:
  input                 Input ROOT file with tuples.

optional arguments:
  -h, --help            show this help message and exit
  --metadata METADATA   JSON input metadata (defaults to <input>.metadata.json)
  --denoms DENOMS       Comma-separated list of efficiency denominators.
  --lines LINES         Comma-separated list of lines to plot results of.
  --vars VARS           Comma-separated list of vars to plot against. Defaults to the parent. Specify a particle by prefixing with the branch name and a colon e.g. "B_s0:PT"
  --decay-name DECAY_NAME
                        Decay short name used in titles.
  --parent PARENT
  --reconstructible-children RECONSTRUCTIBLE_CHILDREN
                        Comma-separated list of child names.
  --true-signal-to-match-to TRUE_SIGNAL_TO_MATCH_TO
                        Comma-separated list of particle names (specified in the annotated decay descriptor) that you'd like to see TrueSim efficiences for.
  --effs-groups EFFS_GROUPS [EFFS_GROUPS ...]
                        Whitespace-separated list of groups of lines for which you'd like to see the inclusive efficiency of, each prefixed with a name and a colon e.g.
                        dimuon_lines:Hlt1DiMuonHighMassLineDecision,Hlt1DiMuonLowMassLineDecision.
  --plot-group-effs     Plot the group efficiencies as well.
  --custom-denoms CUSTOM_DENOMS
                        Comma-separated list of custom cut strings to be applied in addition to the other denominators. E.g. you could pass "Hlt1TrackMVADecision || Hlt1TwoTrackMVADecision" to assess an
                        Hlt2 efficiency with respect to Hlt1. You can also pass each custom denominator a nickname by passing a name with a colon e.g "Hlt1TrackMVAs:Hlt1TrackMVADecision ||
                        Hlt1TwoTrackMVADecision"
  --make-plots
  --legend-header LEGEND_HEADER
                        Give your plot a nice header. TLatex is supported e.g. J/#psi #rightarrow #mu#mu
  --xtitle XTITLE       Optional plot x-axis title. TLatex supported e.g. "p_{T} (B^{0}_{s})"
  --plot-format PLOT_FORMAT
  --plot-prefix PLOT_PREFIX
                        Prefix for the plot file paths.
  --shape-distr-with-denom SHAPE_DISTR_WITH_DENOM
                        For plots where more than 1 denom is shown, specify the denom cut to apply on the shape histogram shown in the background of the plot. For plots with 1 denom only, that denom is
                        used
  --nbins NBINS         Number of efficiency bins.
  --shape-nbins SHAPE_NBINS
                        Number of bins the underlying variable distribution will be plotted in.
  --no-legend
  --json JSON           If set to a filename, write results as json
```

Extra features to explore

2. Tweaking the line: Moore has the nice property that we can instantiate multiple copies of the same line with slightly different thresholds

```
def make_lines():
    standard_lines = [line_builder() for line_builder in all_lines.values()]

    # This is to demonstrate how `configurable`/`bind` works. We could also pass the function a
    with lb0tolcpmum_line.bind(
            name="Hlt2Tutorial_Lb0ToLcpMumNu_LcpToPpKmPip_Pip_pt450MeV_Line",
            pi_pt_min=450 * MeV):
        modified_line = lb0tolcpmum_line()

    return standard_lines + [modified_line]


run_moore(options, make_lines, public_tools)
```

More on this in the Modifying Thresholds section of the Moore documentation.

Extra features to explore

3. Rather than relying on some vaguely representative HLT1 employed in HLT1 filtered MC, you can run HLT1-then-HLT2 and calculate a rate given that any HLT1 line passed. See this section of the documentation.

Be careful with this though, as Allen currently has a passthrough line in the default sequence. So you might want to combine this with `--lines.`

Disclaimer

- TrueSim matching will fail on neutral objects since they have no track. You'll just get DEC efficiencies there.

Summary

- Intro to trigger dataflow: availability of data storage requires analysts to balance HLT efficiency with HLT rate.
- HltEfficiencyChecker is a centralised, automated, easy-to-use and consistent tool, written to facilitate this optimisation for line authors (analysts).
- It has a “beginner” mode (the *wizard*), and an “advanced” by-hand mode. There are a variety of examples of both in MooreAnalysis/HltEfficiencyChecker/options/
- It gives you rates on minimum bias, DEC and TrueSim efficiencies on signal MC. DEC efficiencies will not always give you an accurate picture of the line’s performance.
- You can work HLT2-given-HLT1 efficiencies, and inclusive rates and efficiencies of groups of lines.
- Documentation. Helpdesk: Upgrade HLT2 mattermost. I’ll attach a light version of these slides to the agenda.

Thank you for your attention.
Any questions?

Backup: Writing Decay Descriptors

- Particles: https://gitlab.cern.ch/lhcb-conddb/DDDB/-/blob/master/param/ParticleTable.txt

- Arrows: https://twiki.cern.ch/twiki/bin/view/LHCb/FAQ/LoKiNewDecayFinders#Arrows

Backup: Allen sequences

- Allen is configured in *sequences* - these define the control flow of reconstruction all the way through to selections.

- The default is called hlt1_pp_default, and you don't have to specify it in options (it's the default after all!).

- However, recently the default sequence started using RETINA clusters - AFAIK this is the VELO clustering being done on FPGAs, but these clusters need to be saved in your MC file for the default Allen reconstruction to use them.

- This all happened quite recently, and our files don't have them. We bypass using RETINA clusters with the hlt1_pp_veloSP sequence.

- More on this in the Allen documentation.