Muon Isolation Efficiency vs Pileup

Figures 8.6 & 8.7 of the Muon Isolation Note

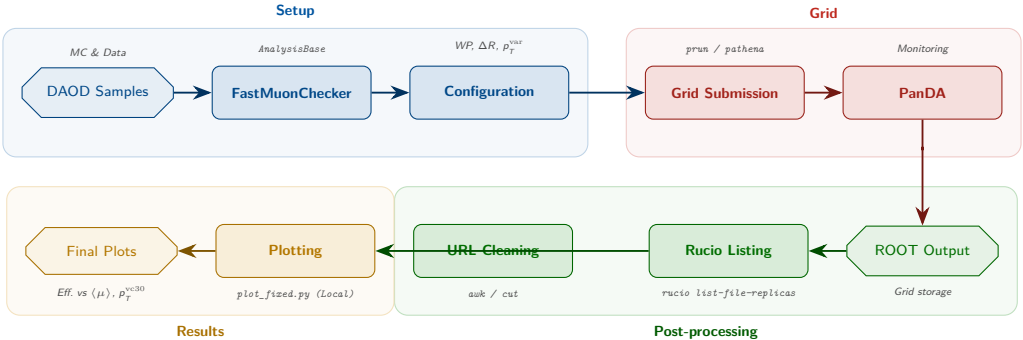
MOHAMMED EL ALLAM

ATLAS Experiment, CERN

June 22, 2026

Outline

Analysis Workflow: From FastMuonChecker to Plotting



Isolation Efficiency vs Pileup

Tag-and-probe on $Z \to \mu\mu$. Per $\langle\mu\rangle$ bin:

$$\varepsilon_{\text{iso}}(\langle\mu\rangle) = \frac{N_{\text{probe}}^{\text{pass WP}}(\langle\mu\rangle)}{N_{\text{probe}}^{\text{total}}(\langle\mu\rangle)}$$

Num. : probes passing iso WP (PFlowTight, Tight, PFlowLoose, Loose)
Den. : all probes in $\langle\mu\rangle$ bin
$\langle\mu\rangle$: `actualInteractionsPerCrossing`

Introduction

Goal: Measure muon isolation efficiency as a function of pileup ($\langle\mu\rangle$) for four working points.

Tag & Probe selection:

- Exactly 2 opposite-sign muons
- $80 < m_{\mu\mu} < 100\,\text{GeV}$ (Z peak)

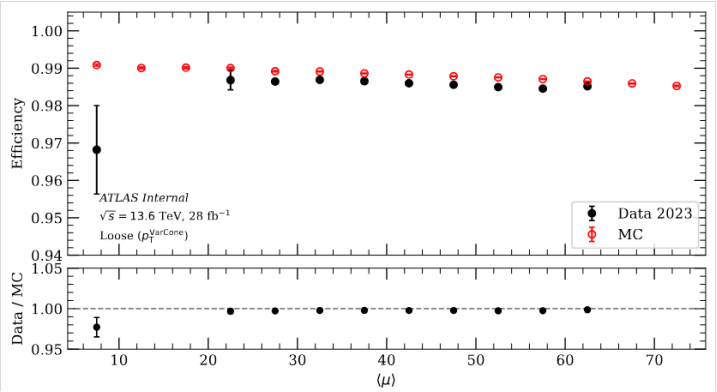
Technical details

- Software: AnalysisBase,25.2.89
- ROOT Version: 6.36.04
- Package:fastMuonChecker_modified.py

Four isolation working points:

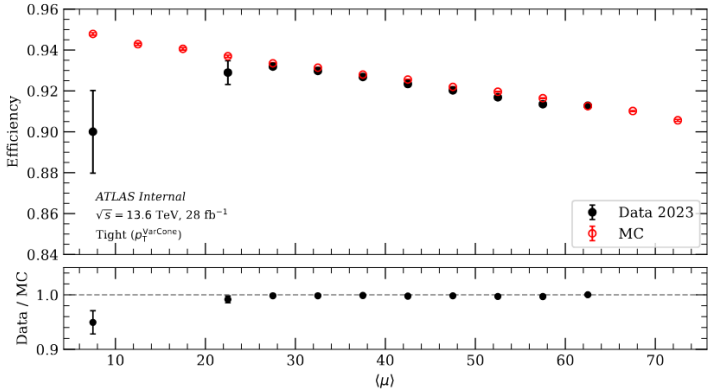
WP	Variables
Loose_VarRad	Track + Calo
Tight_VarRad	Track + Calo
PflowLoose_VarRad	Track + PFlow
PflowTight_VarRad	Track + PFlow

Isolation Efficiency vs Pileup – Loose_VarRad



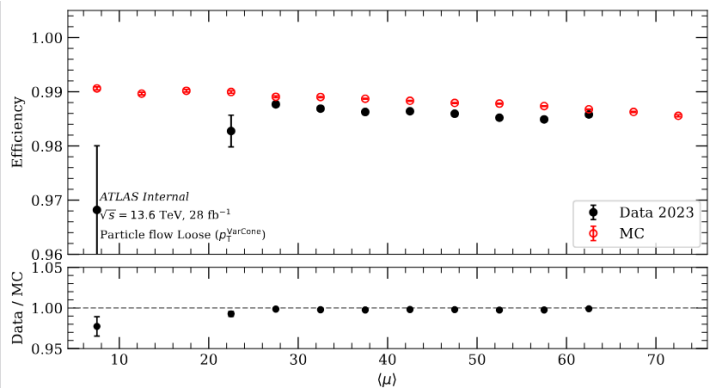
Working Point:
Loose_VarRad

Isolation Efficiency vs Pileup – Tight_VarRad



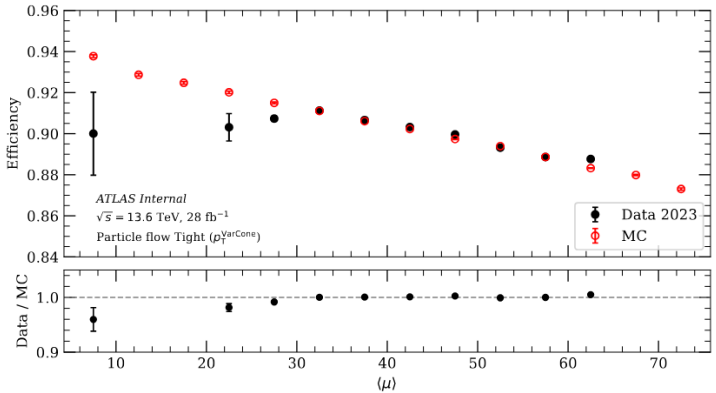
Working Point:
Tight_VarRad

Isolation Efficiency vs Pileup – PflowLoose_VarRad



Working Point:
PflowLoose_VarRad

Isolation Efficiency vs Pileup – PflowTight_VarRad



Working Point:
PflowTight_VarRad

Back Up

```
Message from root@lxplus956.cern.ch on <no tty> at 16:23 ...
Dear LxPlus User melallam

The process (python) has been killed as the CGroup of all
your processes has reached your memory allocation.

Memory cgroup out of memory: Killed process 2196655 (python) total-vm:63087636kB, anon-rss:35870920kB, file-rss:9720kB, shmem-rss:0kB, UID:191311 pgta
bles:75884kB oom_score_adj:0

To check your memory limits of your CGroup and current usage:

systemctl status user-0.slice
EOF
Killed
```

Figure: Out-of-memory termination of the plotting process on `lxplus` due to CGroup memory limits.

Processing Statistics

Metric	Count	%
Total files	3350	100%
Processed successfully	1686	**50.3%**
Failed	1664	49.7%

Table: Summary of file processing results

Overview of Modifications

Source Code

- **Modified:** `fastMuonChecker_modified.py`
+ **Original:** `fastMuonChecker.py`

7 Major Categories

1. **Data/MC Separation** — `-isData`, `-year` flags
2. **Multiple Isolation WPs** — 1 WP → 4 WPs
3. **Removed Variables** — dimuon, jets, tracks, flags
4. **Truth Muon Handling** — auto-detect container
5. **Muon Selection Logic** — no isolation cut
6. **Safe Auxiliary Access** — `safe_auxdata()`
7. **Misc. Changes** — PRW, avgMu, types

Legend

+ Added in modified version
− Removed from original
∼ Changed / Modified

Key Philosophy Change

Original: applies isolation cut → stores only passing muons
Modified: stores **all** muons with per-WP pass/fail flags

1. Data/MC Separation

− Original: No Data/MC flag

```
parser.add_argument("-r", "--run", ...)
parser.add_argument("-m", "--maxEvents", ...)
parser.add_argument("-v", "--verbose", ...)
# No --isData or --year argument
# MC tools always initialized (crash on data)
iff_classifier = tools.get_TruthClassificationTool()
muon_eff_sf_tool = tools.get_MuonEfficiency...
muon_iso_sf_tool = tools.get_MuonEfficiency...
muon_ttva_sf_tool = tools.get_MuonEfficiency...
xsec_tool = tools.get_PMGCrossSectionTool()
prw_tool = tools.get_PileupReweightingTool(...)
```

+ Modified: Conditional MC tools

```
parser.add_argument("--isData",
action="store_true")
parser.add_argument("--year", default="2022",
choices=["2022", "2023"])

if not args.isData:
iff_classifier = tools.get_Truth...
muon_eff_sf_tool = tools.get_Muon...
muon_iso_sf_tools = {}   # dict per WP
for wp in isolation_wps:
muon_iso_sf_tools[wp] = ...
muon_ttva_sf_tool = tools.get_Muon...
xsec_tool = tools.get_PMGCrossSection...
prw_tool = tools.get_PileupReweighting...
```

Impact: Code can now run on **real data** without crashing on missing truth/MC containers.
All MC-only branches (`dsid`, `mcEventWeight`, `pileupWeight`, truth muons, SFs) are conditionally created.

2. Multiple Isolation Working Points

− Original: Single WP

```
# One isolation tool
muon_isolation_tool = tools.get_IsolationSe-
lectionTool(MuonWP="Loose_VarRad")

# One SF tool
muon_iso_sf_tool = tools.get_MuonEfficiency-
ScaleFactors("MuonIsolationScaleFactors",
WorkingPoint="Loose_VarRadIso", ...)

# One branch
muon_iso_SF = ROOT.std.vector("float")()
nt.Branch("muon_iso_SF", muon_iso_SF)

# In muon loop: cut on isolation
passes_iso = muon_isolation_tool.accept(...)
if not passes_iso:
continue  # SKIP muon!
```

+ Modified: 4 WPs

```
isolation_wps = ["Loose_VarRad",
"Tight_VarRad",
"PflowLoose_VarRad",
"PflowTight_VarRad"]
isolation_tools = {}
for wp in isolation_wps:
isolation_tools[wp] = tools.get_Isolation-
SelectionTool(MuonWP=wp, name=f"...")

# Per-WP branches
muon_passIso = {}
muon_iso_SF = {}
for wp in isolation_wps:
muon_passIso[wp] = ROOT.std.vector("bool")()
muon_iso_SF[wp] = ROOT.std.vector("float")()

# In muon loop: store all, no cut
for wp in isolation_wps:
muon_passIso[wp].push_back(
bool(isolation_tools[wp].accept(...)))
```

Impact: Enables isolation efficiency studies for **all 4 WPs simultaneously** without re-running.
No isolation cut applied → denominator preserved for efficiency calculation.

3. Removed Variables & Branches

– Removed from Original

Efficiency flags (now handled at plotting level):

```
muon_passes_medium = ROOT.std.vector("char")()
muon_passes_iso    = ROOT.std.vector("char")()
muon_passes_ttva   = ROOT.std.vector("char")()
```

Isolation variable (moved to plotting):

```
muon_ptvarcone30 = ROOT.std.vector("float")()
```

Dimuon invariant mass calculation:

```
dimuon_mass     = ROOT.std.vector("double")()
dimuon_mu1_idx = ROOT.std.vector("int")()
dimuon_mu2_idx = ROOT.std.vector("int")()
```

ΔR **to closest jet**:

```
muon_deltaR_closest_jet = ROOT.std.vector("float")()
# + jet container access (AntiKt4EMTopoJets)
```

Inner detector tracks:

```
track_pt, track_eta, track_phi
track_ptvarcone30
# + InDetTrackParticles container access
```

Rationale

- ∼ **Efficiency flags**: redundant — selection is already applied; efficiency computed at plotting stage from pass/fail counts
- ∼ **ptvarcone30**: can be recomputed from stored tracks at analysis level
- ∼ **Dimuon mass**: computed at plotting level from stored 4-vectors; reduces ntuple size
- ∼ ΔR **to jet**: analysis-specific; not needed in base ntuple
- ∼ **ID tracks**: large container → significant disk usage; moved to dedicated studies

Result

12 branches removed → smaller output ROOT files
→ faster grid jobs, less storage on EOS

4. Truth Muon Handling

− Original: Hardcoded container

```
# Always uses MuonTruthParticles
for truth_muon in mgr.eventTree()
.MuonTruthParticles:
truth_muon_pt.push_back(truth_muon.pt())
...
# Direct auxdata access (can crash)
truth_muon_classifierParticleOrigin
.push_back(truth_muon
.auxdataConst'truthOrigin')

# Truth-reco link: direct vector assignment
truth_muon_recoLink[index] = muon_pt.size()-1
```

+ Modified: Auto-detect + safe access

```
# Auto-detect truth container
truth_muon_container_name = None
for _candidate in ['MuonTruthParticles',
'TruthMuons']:
if hasattr(_tree, _candidate):
truth_muon_container_name = _candidate
break
if truth_muon_container_name is None \
and hasattr(_tree, 'TruthParticles'):
truth_muon_container_name = 'TruthParticles'

# Safe auxdata with fallback
val = safe_auxdata(truth_muon, 'unsigned int',
'classifierParticleOrigin')
if val is None:
val = safe_auxdata(truth_muon,
'unsigned int', 'truthOrigin', 0)

# Reco link via Python list (safe assignment)
truth_muon_recoLink_tmp = []  # per event
truth_muon_recoLink_tmp.append(-1)
# ... later:
truth_muon_recoLink_tmp[index] = int(...)
```

Impact: Works with **any** MC sample (Pythia/Herwig/Sherpa) regardless of truth container naming convention.
No crashes on missing auxdata decorations — graceful fallback to alternative names.

5. Muon Selection Logic

− Original: Cuts on isolation + flags

```
# 1) Medium selection + flag
passes_medium = muon_selection_tool.accept(...)
muon_passes_medium.push_back(
1 if passes_medium else 0)
if not passes_medium:
continue

# 2) Isolation cut (REMOVES muons!)
passes_iso = muon_isolation_tool.accept(...)
muon_passes_iso.push_back(
1 if passes_iso else 0)
if not passes_iso:
continue   # <-- muon lost!

# 3) TTVA cut + flag
passes_ttva = ...
muon_passes_ttva.push_back(
1 if passes_ttva else 0)
if not passes_ttva:
continue
```

+ Modified: No isolation cut

```
# 1) Medium selection only
if not muon_selection_tool.accept(muon_to_use):
continue

# 2) TTVA cut (kept)
if abs(d0/d0_sigma) > 3 or \
abs(abs(z0+vz-z)*sinTheta) > 0.5:
continue

# 3) Store ALL muons that pass Medium+TTVA
#    Isolation is ONLY stored as flags:
for wp in isolation_wps:
muon_passIso[wp].push_back(
bool(isolation_tools[wp].accept(...)))

# No muons lost to isolation cut!
# Efficiency = N_pass / N_total per <mu> bin
```

Impact: Preserves the **full denominator** for isolation efficiency calculation:
$\varepsilon_{\mathrm{iso}}(\langle\mu\rangle) = N_{\mathrm{pass}}^{\mathrm{WP}}/N_{\mathrm{total}}$ per pileup bin — impossible with original code (biased denominator).

6. Safe Auxiliary Data Access

− Original: Direct access (can crash)

```
# No safe_auxdata function

# Direct access --- crashes if missing:
muon_truthOrigin.push_back(
muon_to_use.auxdataConst['int']
('truthOrigin'))

muon_truthType.push_back(
muon_to_use.auxdataConst['int']
('truthType'))

# Truth link --- no try/except:
truth_link = muon_to_use.auxdataConst[
"ElementLink<DataVector<xAOD::
TruthParticle_v1> >"
]("TruthLink")
```

+ Modified: Robust access

```
def safe_auxdata(particle, dtype, name,
default=None):
"""Safely access auxdataConst."""
try:
return particle.auxdataConstname
except:
return default

# Safe access with fallback:
muon_truthOrigin.push_back(
safe_auxdata(muon_to_use, 'int',
'truthOrigin', -1))

# Truth link with try/except:
try:
truth_link = muon_to_use.auxdataConst[
"ElementLink<...>"
]("TruthLink")
except:
truth_link = None
```

Impact: No more **runtime crashes** on samples with missing/renamed decorations.
Default value −1 flags missing data for downstream analysis.

7. Miscellaneous Changes

~ PRW Tool: `apply()` argument

```
# Original:
prw_tool.apply(ei, False)
```

```
# Modified:
prw_tool.apply(ei, True)
```

+ New branch: `avgMu`

```
avgMu = np.array([0.], dtype=np.float32)
nt.Branch("avgMu", avgMu, "avgMu/F")
# Filled per event:
avgMu[0] = ei.averageInteractionsPerCrossing()
```

+ New branch: `isData`

```
isData_val = np.array(
[1 if args.isData else 0], dtype=np.int32)
nt.Branch("isData", isData_val, "isData/I")
```

~ Trigger matched: type change

```
# Original: char (0/1) + explicit flag
muon_is_trigger_matched =
ROOT.std.vector("char")()
muon_is_trigger_matched.push_back(
1 if matched else 0)
```

```
# Modified: bool + for/else pattern
muon_is_trigger_matched =
ROOT.std.vector("bool")()
for trigger in passed_triggers:
if tdt.isPassed(trigger) and \
matchtool.match(...):
muon_is_trigger_matched.push_back(True)
break
else:
muon_is_trigger_matched.push_back(False)
```

~ Metadata tree: conditional

```
# Original: always created
mt = ROOT.TTree("metadata","metadata")
mt.Fill(); of.WriteTObject(mt)
```

```
# Modified: only for MC
if not args.isData:
mt = ROOT.TTree("metadata","metadata")
...
mt.Fill(); of.WriteTObject(mt)
```

Summary of All Modifications

Feature	Original	Modified	Motivation
`-isData` flag	−	+	Run on real data
`-year` flag	−	+	Year-specific configs
Isolation WPs	1 (Loose)	4 (all)	Multi-WP efficiency study
Isolation cut in loop	+	−	Preserve denominator
`safe_auxdata()`	−	+	No runtime crashes
Truth container auto-detect	−	+	Pythia/Herwig/Sherpa compat.
`avgMu` branch	−	+	Pileup-dependent studies
`isData` branch	−	+	MC/Data flag in ntuple
`muon_ptvarcone30`	+	−	Moved to plotting
`dimuon_mass`	+	−	Computed at plotting
`deltaR_closest_jet`	+	−	Not needed in base ntuple
ID track branches	+	−	Too large; dedicated study
Efficiency flags	+	−	Redundant with selection
`recoLink` via Python list	−	+	Safe index assignment
PRW `apply(ei, ...)`	`False`	`True`	Correct MC treatment
Trigger matched type	`char`	`bool`	Cleaner type
Metadata tree	Always	MC only	No empty tree for data

Net result: Leaner, more robust ntuple production — supports Data + MC, all 4 isolation WPs, and any generator truth format.

Data Samples: data23_13p6TeV – physics_Main DAOD_PHYS

Run 451094

- r14858_p5785_p6269
- r14858_p5785_p6482
- r15774_p6304_p6482
- r15774_p6304_p6700
- r15774_p6304_p7019
- r15774_p6304_p7267

Run 455924

- f1367_m2185_p5858
- r14858_p5785_p6269
- r14858_p5785_p6482
- r15774_p6304_p6482
- r15774_p6304_p6700
- r15774_p6304_p7019
- r15774_p6304_p7267

Run 455975

- f1367_m2185_p6269
- f1367_m2185_p6482
- r15774_p6304_p6482
- r15774_p6304_p6700
- r15774_p6304_p7019
- r15774_p6304_p7267

Run 456749

- f1370_m2191_p6269
- f1370_m2191_p6482
- r15774_p6304_p6482
- r15774_p6304_p6700
- r15774_p6304_p7019
- r15774_p6304_p7267

Monte Carlo Samples: mc23_13p6TeV

DSID 601190 – PhPy8EG_AZNLO_Zmumu (DAOD_PHYS)

- e8514_e8528_s4159_s4114_r15513_r15514_p6026
- e8514_s4159_r15513_p6026