

Introduction to DAVINCI

- Introduction
 - overview of DAVINCI structure
 - Input
- My first Selection
 - Read $J/\psi \rightarrow \mu\mu$ all in python, no C++ to write

LHCb Week
3 December 2010

Patrick Koppenburg



DAVINCI LINKS

- DAVINCI web page:

<http://cern.ch/LHCb-release-area/DOC/davinci/>

From there you'll find :

- Some documentation. Links to doxygen.
- The [DAVINCI wiki](#) and [Tutorial page](#)
- Bugs and feature requests are tracked in the [LHCb software Savannah page](#). If you think you see a bug, go there first.
- Any DAVINCI question can be asked at the DAVINCI mailing list: lhcb-davinci@cern.ch .
 - You need to be registered to use it. You can do that online.
- Distributed analysis question should be asked at lhcb-distributed-analysis@cern.ch .
- General software questions should go to lhcb-soft-talk@cern.ch

Always give all information that is relevant to your question! (no “it stopped working”)

DAVINCI



INPUT: ProtoParticles

ProtoParticles : are the end of the reconstruction stage

- have all the links about how they have been reconstructed
- have a list of PID hypothesis with a probability
- contain the *kinematic* information

CHARGED ProtoParticles : One per Track

→ Add Rich, Calo, Muon info

NEUTRAL ProtoParticles : One per Cluster

→ Add Track info (could be an electron)

Particles

- Particle = ProtoParticle + one PID *choice*
 - ➔ one defined mass
- Physics analyses deal with Particles
 - You need to know the 4-vectors to compute the mass of a resonance
- The PID is your choice
 - The same ProtoParticle can be made as a π and as a μ ...
 - This makes sense. Think of a pion from $B \rightarrow \pi\pi$ decaying in flight. Does it stop being a signal pion because it decayed before the Muon detector?
 - Some ProtoParticles can be ignored
 - All this is done by configuring a ParticleMaker algorithm
 - You don't need to worry about the configuration.
 - Many standards are pre-defined
 - But you need to choose which to use
 - ➔ Next slide

STANDARD Particles

- The Particles are actually already done for you. To ensure that everybody agrees on what is a K^+ , a π or a K_S^0 , we have a set of standard particles predefined.
- They are defined in `python/CommonParticles/*.py` in the `Phys/CommonParticles` package.
- All you need to know are the names of the algorithm that created them : `StdLooseKaons`, `StdTightProtons` ...

STDNOPIDsXXXX: All tracks are made to XXXX

STDLOOSEXXXX: Loose PID cuts for hypothesis XXXX (no cuts for pions)

STDTIGHTXXXX: Tight PID cuts for hypothesis XXXX

HOW TO SELECT PARTICLES

There are several ways to quickly get a physics result:

PLAIN C++: DVAlgorithm inherits from GaudiAlgorithm (and GaudiTupleAlg ...), some typing is saved

LoKi: “loops and kinematics”. Templated C++. More typing saved.

GAUDIPYTHON, BENDER: Interactive python.

LoKi::Hybrid: Used in Stripping.

The common assumption is that physicists always do the same, hence any line of C++ you type is likely to be a duplication of what your office-mate is typing right now.

GENERIC SELECTION FRAMEWORK

Any (flavour-) physics analysis is a sequence of $A \rightarrow B C (\dots)$, with some cuts in between.

Hence: An analysis algorithm should know:

Anything else?

GENERIC SELECTION FRAMEWORK

Any (flavour-) physics analysis is a sequence of $A \rightarrow B C (\dots)$, with some cuts in between.

Hence: An analysis algorithm should know:

- Where to get the particles

- Where to put the data

Handled by DVAlgorithm

Anything else?

GENERIC SELECTION FRAMEWORK

Any (flavour-) physics analysis is a sequence of $A \rightarrow B C (\dots)$, with some cuts in between.

Hence: An analysis algorithm should know:

- Where to get the particles
- What decay to reconstruct
- Where to put the cuts

Anything else?

```
for { LHCb::Particle::ConstVector::const_iterator mK = KMinus.begin() ;  
      mK != KMinus.end() ; ++mK ){  
  for { LHCb::Particle::ConstVector::const_iterator pK = KPlus.begin()  
        pK != KPlus.end() ; ++pK ){  
    for { LHCb::Particle::ConstVector::const_iterator pi = Pions.begin()  
          pi != Pions.end() ; ++pi ){  
      [...]    }  
  }  
}
```

Can be shorter:

```
for ( Loop Ds = loop( "K K pi", "D_s+", FitVertex ) ; Ds ; ++Ds ) {
```

Or:

```
DsForBs2DsPi.DecayDescriptor = "[D_s+ -> K+ K- pi+]cc"
```

GENERIC SELECTION FRAMEWORK

Any (flavour-) physics analysis is a sequence of $A \rightarrow B C (\dots)$, with some cuts in between.

Hence: An analysis algorithm should know:

- Where to get the particles
- What decay to reconstruct
- What cuts to apply
- Where to put the data

Hard-coding cuts is a bad idea . . .

Better to use options of the algorithm

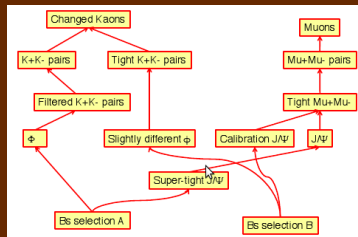
. . . or predefined filters configurable by options:

```
DsForBs2DsPi.MotherCut = "PT > 2000*MeV"
```

Anything else?

THE SELECTION FRAMEWORK

- A selection is a sequence of algorithms reading in `Particles` and writing out other `Particles`
 - You want to make sure that the output of one algorithm is the input to another one
- Some of this is enforced in `C++`, but not in `python`.
- ✗ You can make mistakes



Use a selection framework that tells you from the start that something is wrong

FilterDesktop

Sometimes it is helpful to filter what is on the TES:

```
_jspiFilter = FilterDesktop("_jspiFilter")
_jspiFilter.Code = '(PT>1*GeV) &
                    (MINTREE(ABSID=='mu+',P)>8*GeV)'

# make a Selection
JpsiFilterSel = Selection(name = 'JpsiFilterSel',
                          Algorithm = _jspiFilter,
                          RequiredSelections = [ JpsiSel ])
# JpsiSel is the input to _jspiFilter
```

→ Cuts on the momentum of the muons and on the p_T of the mother.

DEMO!

```
from math import sqrt
from LoKiPhys.decorators import *
from LoKiCore.functions import monitor
p = LHCb.Particle()
p.setParticleID( LHCb.ParticleID(11) )
m = p.momentum()
m.SetPx ( 1000 )
m.SetPy ( -1000 )
m.SetPz ( 10000 )
m.SetE ( sqrt( m.P2() + 5000*5000 ) )
p.setMomentum ( m )
fun = PX+PY
print PX(p), PY(p) , fun(p)
fun2 = PX>750
print fun2(p)
fun3 = monitor(fun2)
print fun3(p)
from LoKiCore.doxygenurl import browse
browse(PT)
```

Stolen from Vanya

Read J/ψ 's

WHAT TO DO?

The workflow of you analysis should be:

- ① Read your candidates
 - ② Refine your candidates (more cuts)
 - ③ TisTos your candidate
 - If TIS, you're lucky → keep it
 - If TOS, keep it only if Tossed by one of "your" lines
 - ④ Store (or fit) it somehow
 - ① DST
 - ② MicroDST (better)
 - ③ DecayTreeTuple (not good, but sometimes useful)
- Make sure you know what you need before starting

THE DATA

- Reco08-Stripping12b-Merged is now being produced
 - ➔ One file per run (if size permits)
- All data will be reprocessed by beginning of January. If you need more stats look at Reco07-Stripping11-Merged (down) and Reco06-Stripping10-Merged (up)
- Never use the data in locations without -Merged

The screenshot shows the 'File Settings' window with a tree view of data files and event types. The tree is organized into several categories:

- Processing Pass**
 - Real Data
 - Real Data + Reco03
 - Real Data + Reco04
 - Real Data + Reco04-Stripping05
 - Real Data + Reco04-Stripping05-PrescaledMinBias
 - Real Data + Reco04-Stripping07
 - Real Data + Reco04-Stripping07-Merged
 - Real Data + Reco05
 - Real Data + Reco05-Stripping09
 - Real Data + Reco05-Stripping09-Merged
 - Real Data + Reco05-Stripping09-Prescaled
 - Real Data + Reco05-Stripping09-Prescaled-Merged
 - Real Data + Reco06
 - Real Data + Reco06-Stripping10
 - Real Data + WF-Validation-Reco06-Stripping10-2
 - Real Data + WS-Validation-Reco06
 - Reco06-Stripping10-Merged
- Event types**
 - File types
 - SHADRON DST
 - CALIBRATION DST
 - CHARM DST
 - DIELECTRON DST
 - DIMUON DST
 - No. of Files/Events** (287/6580806)
 - EW DST
 - MINIBIAS DST
 - RADIATIVE DST
 - SEMILEPTONIC DST
 - SETC
 - WF-Validation-Reco06-Stripping10-2-Merged
 - Beam3500GeV-VeloClosed-MagUp-Excl-MU
 - Beam3500GeV-VeloClosed-MagUp-Excl-R1-R2
 - Beam3500GeV-VeloOpen-MagDown
 - Beam3500GeV-VeloOpen-MagDown-Excl-IT-R1-R2-TT
 - Beam3500GeV-VeloOpen-MagDown-Excl-IT-R1-R2-TT-VF

At the bottom, the 'Queries' section is visible, with the following options:

- SimCond/ProcessingPass/EventType/Production/FileType/ProgramFiles
- Event type/SimCond/ProcessingPass/Production/FileType/ProgramFiles
- Production lookup
- Run lookup

HOW TO READ YOUR CANDIDATES

I want the candidates of the tight $J/\psi \rightarrow \mu\mu$ selection

① Let's look at the Dimuon stream. Do:

```
SetupProject DaVinci
python
from StrippingSettings.Stripping12.StreamDimuon import stream
locations = {}
for line in stream.lines :
    locations[ line.name() ] = stream.name() + '/' + line.outputLocation()

for line, loc in locations.iteritems() :
    if 'DiMuon' in line : print line, loc
...
StrippingNeuroBayesJPsiLine Dimuon/Phys/NeuroBayesJPsiLine
StrippingNeuroBayesMuMuLine Dimuon/Phys/NeuroBayesMuMuLine
StrippingDiMuonHighMassSameSignLine Dimuon/Phys/DiMuonHighMassSameSignLine
StrippingBd2KstarMuMu_SignalTriggered Dimuon/Phys/Bd2KstarMuMu_SignalTriggered
StrippingBs2MuMuNoMuIDLooseLine Dimuon/Phys/Bs2MuMuNoMuIDLooseLine
```

➔ Line StrippingNeuroBayesMuMuLine puts candidates in
/Event/Dimuon/Phys/NeuroBayesMuMuLine

See <https://twiki.cern.ch/twiki/bin/view/LHCb/LHCbStripping>

HOW TO READ YOUR CANDIDATES

I want the candidates of the tight $J/\psi \rightarrow \mu\mu$ selection

- 1 Line StrippingNeuroBayesMuMuLine puts candidates in
`/Event/Dimuon/Phys/NeuroBayesMuMuLine`
- 2 Actually, the candidates are moved there from
`/Event/Phys/NeuroBayesMuMuLine`

See <https://twiki.cern.ch/twiki/bin/view/LHCb/LHCbStripping>

HOW TO READ YOUR CANDIDATES

I want the candidates of the tight $J/\psi \rightarrow \mu\mu$ selection

- 1 Line StrippingNeuroBayesMuMuLine puts candidates in `/Event/Dimuon/Phys/NeuroBayesMuMuLine`
- 2 Actually, the candidates are moved there from `/Event/Phys/NeuroBayesMuMuLine`
- 3 Try `PrintDecayTree` (here with $B \rightarrow K^*\gamma$):

```
from Configurables import DaVinci, PrintDecayTree
DaVinci().appendToMainSequence( [ PrintDecayTree(
    InputLocations = [ '/Event/Radiative/Phys/SelBd2KstGamma' ] ) ] )
DaVinci().DataType = "2010"
```

```
PrintDecayTree.PrintDecayINFO
<----- Particle ----->
      Name          E          M          P          Pt          phi          Vz
                   MeV       MeV       MeV       MeV       mrad       mm
BO
+-->K*(892)0      32923.64    889.98    32911.61    1560.75    2828.35    38.45
|+-->K+           17020.85    493.68    17013.69    905.13     2539.99    37.14
|+-->pi-          15902.98    139.57    15902.37    740.29    -3099.25    37.04
+-->gamma         15014.49     0.00    15014.49    2235.83    -721.99     0.00
```

See <https://twiki.cern.ch/twiki/bin/view/LHCb/LHCbStripping>

HOW TO READ YOUR CANDIDATES

I want the candidates of the tight $J/\psi \rightarrow \mu\mu$ selection

- 1 Line StrippingNeuroBayesMuMuLine puts candidates in `/Event/Dimuon/Phys/NeuroBayesMuMuLine`
- 2 Actually, the candidates are moved there from `/Event/Phys/NeuroBayesMuMuLine`
- 3 Try `PrintDecayTree` (here with $B \rightarrow K^*\gamma$):
- 4 What if I have no candidates? Still stick to you line!

You can check the result of the selection with

```
from Configurables import LoKi_HDRFilter as StripFilter
MySequencer = GaudiSequencer('Sequence')
MySequencer.Members += [ StripFilter( 'StripPassFilter',
                                     Code="HLT_PASS('StrippingNeuroBayesMuMuLineDecision')",
                                     Location="/Event/Strip/Phys/DecReports" ) ]
```

Note the location (the default is the HLT).

See <https://twiki.cern.ch/twiki/bin/view/LHCb/LHCbStripping>

HOW TO REFINE YOUR CANDIDATES

```
from Gaudi.Configuration import *
line = 'NeuroBayesMuMuLine'
location = '/Event/Dimuon/Phys/'+line
from Configurables import DaVinci

# get classes to build the SelectionSequence
from PhysSelPython.Wrappers import AutomaticData, Selection, SelectionSequence
# Get the Candidates from the DST.
# AutomaticData is for data on the DST
JpsiSel = AutomaticData(Location = location)
# Filter the Candidate.
from Configurables import FilterDesktop
_jpsiFilter = FilterDesktop('jpsiFilter', Code = 'MM<4000')

# make a Selection
JpsiFilterSel = Selection(name = 'JpsiFilterSel',
                        Algorithm = _jpsiFilter,
                        RequiredSelections = [ JpsiSel ])

# build the SelectionSequence
JpsiSeq = SelectionSequence('SeqJpsi',
                          TopSelection = JpsiFilterSel,
                          )

DaVinci().appendToMainSequence( [ JpsiSeq.sequence() ] )
```

See <https://twiki.cern.ch/twiki/bin/view/LHCb/ParticleSelection>

HOW TO REFINE YOUR CANDIDATES

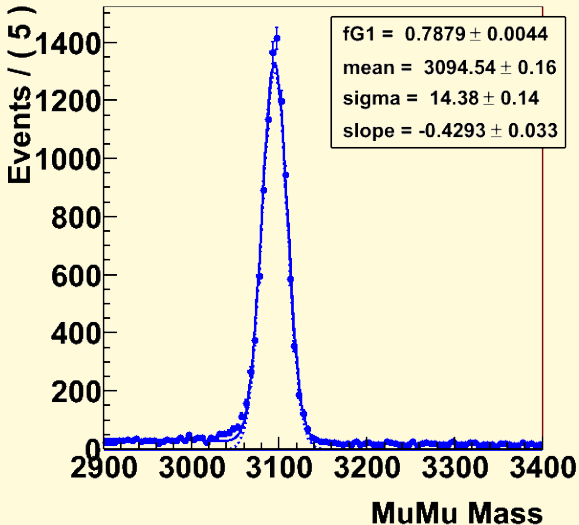
```
from Gaudi.Configuration import *
line = 'NeuroBayesMuMuLine'
location = '/Event/Dimuon/Phys/'+]
from Configurables import DaVinci

# get classes to build the Selection
from PhysSelPython.Wrappers import
# Get the Candidates from the DST.
# AutomaticData is for data on the
JpsiSel = AutomaticData(Location =
# Filter the Candidate.
from Configurables import FilterDe
_jpsiFilter = FilterDesktop('jpsiF

# make a Selection
JpsiFilterSel = Selection(name = '
                        Algorith
                        Requirec

# build the SelectionSequence
JpsiSeq = SelectionSequence('SeqJf
                        TopSe
                        )

DaVinci().appendToMainSequence( [
```



See [https://twiki.cern](https://twiki.cern.ch/twiki/bin/view/LHCb/NeuroBayes)

HOW TO SELECT A TRIGGER LINE

I assume you already know which L0/Hlt1/Hlt2 lines you want

- You can use the DecReports to refine your selection

```
from Configurables import LoKi_HDRFilter as HltFilter
MySequencer = GaudiSequencer('Sequence')
MySequencer.Members += [ HltFilter( 'HltPassFilter',
                                  Code="HLT_PASS('Hlt1Photon.*Decision')" ]
```

See <https://twiki.cern.ch/twiki/bin/view/LHCb/TriggerTisTos>

HOW TO SELECT A TRIGGER LINE

I assume you already know which L0/Hlt1/Hlt2 lines you want

- You can use the `DecReports` to refine your selection
- But to really understand what the trigger is doing to your signal you need to `TisTos` it.
 - See [Tomasz' tutorial](#)
 - If you use `DecayTreeTuple`, configure `TupleToolTISTOS`

See <https://twiki.cern.ch/twiki/bin/view/LHCb/TriggerTisTos>

```
##### DecayTreeTuple
from Configurables import DecayTreeTuple, TupleToolTrigger,
tuple = DecayTreeTuple("Jpsi_Tuple")
tuple.ToolList += [
    "TupleToolGeometry"
    , "TupleToolKinematic"
    , "TupleToolPrimaries"
    , "TupleToolEventInfo"
    , "TupleToolTrackInfo"
    , "TupleToolTISTOS"
    , "TupleToolAngles"
    , "TupleToolPid"
    , "TupleToolPropertime"
]
tuple.Decay = "J/psi(1S) -> ^mu+ ^mu-"
tuple.InputLocations = [ pt, JpsiSeq.outputLocation() ]
tuple.addTool(TupleToolTISTOS)
tuple.TupleToolTISTOS.TriggerList = [ "Hlt2DiMuonUnbiasedJP"
tuple.TupleToolTISTOS.VerboseHlt2 = True
#####

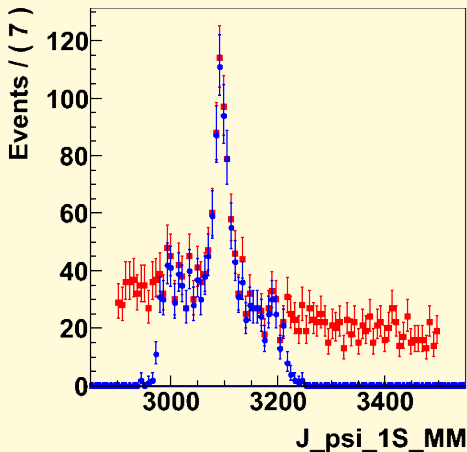
DaVinci().appendToMainSequence( [ JpsiSeq.sequence(), tuple

DaVinci().DataType = "2010"
DaVinci().EvtMax = -1
DaVinci().PrintFreq = 100
DaVinci().TupleFile = "Jpsi.root"
```

HOW TO SELECT A TRIGGER LINE

This shows the dimuon mass

- For B candidates that are Hlt2Hlt2DiMuon-UnbiasedJPsi triggered
 - TOS
 - all

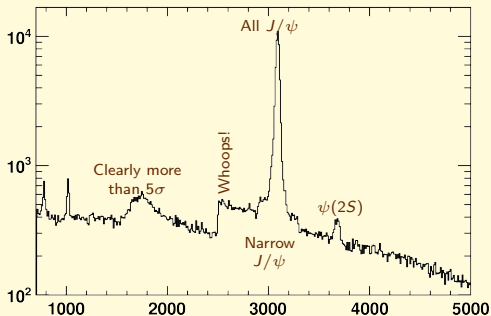


See <https://twiki.cern.ch/twiki/bin/view/LHCb/TriggerTisTos>

OBSERVATION OF A RESONANCE AT 1.8 GEV

This is the dimuon spectrum on the dimuon stream with $\mu\mu$ combinatorics redone

It is biased by all selections. Including “trivial” ones (thresholds) and complicated one, like $B \rightarrow \mu\mu K^*$.



NEVER REDO COMBINATORICS ON A STREAM!

You will not be able to unfold the biases

ALWAYS USE THE CANDIDATES!

The only exception is the MB stream

NEVER RUN ON SEVERAL STREAMS MIXED-UP!

Why would you do that?

EXERCISES!

Ex. 0: **Set things up**

Ex. 1: Loop over muons and make some plots

Ex. 2: Extend the algorithm to make a J/ψ (if you have time)

Ex. 3: Make your algorithm more generic: select also a ϕ

Do Ex. 1 to 3 if you plan to develop C++ in DAVINCI.

Ex. 4: The recommended way of writing a selection

- Everything you need is on the wiki page
- The main difficulty is to figure out what to copy-paste where.
- Don't be afraid to ask if you are unsure

Ex. 5: Debugging

Ex. 6: MC truth, Trigger, Tagging, and much more

Ex. 7: More Tuples

Ex. 8: **Read Stripped DSTs**



Backup

Standards:

- Standard Code
- Cuts
- More

Exercises are provided in the Tutorial/Analysis package.



HOW TO SELECT PARTICLES

There are several ways to quickly get a physics result:

PLAIN C++: DVAlgorithm inherits from GaudiAlgorithm (and GaudiTupleAlg ...), some typing is saved

LoKi: “loops and kinematics”. Templated C++. More typing saved.

GAUDIPYTHON, BENDER: Interactive python.

LoKi::Hybrid: Used in Stripping.

The common assumption is that physicists always do the same, hence any line of C++ you type is likely to be a duplication of what your office-mate is typing right now.

GENERIC SELECTION FRAMEWORK

Any (flavour-) physics analysis is a sequence of $A \rightarrow B C (\dots)$, with some cuts in between.

Hence: An analysis algorithm should know:

Anything else?

GENERIC SELECTION FRAMEWORK

Any (flavour-) physics analysis is a sequence of $A \rightarrow B C (\dots)$, with some cuts in between.

Hence: An analysis algorithm should know:

- Where to get the particles

- Where to put the data

Handled by DVAlgorithm

Anything else?

GENERIC SELECTION FRAMEWORK

Any (flavour-) physics analysis is a sequence of $A \rightarrow B C (\dots)$, with some cuts in between.

Hence: An analysis algorithm should know:

- Where to get the particles
- What decay to reconstruct
- Where to put the cuts

Anything else?

```
for { LHCb::Particle::ConstVector::const_iterator mK = KMinus.begin() ;  
      mK != KMinus.end() ; ++mK ){  
  for { LHCb::Particle::ConstVector::const_iterator pK = KPlus.begin()  
        pK != KPlus.end() ; ++pK ){  
    for { LHCb::Particle::ConstVector::const_iterator pi = Pions.begin()  
          pi != Pions.end() ; ++pi ){  
      [...]    }  
  }  
}
```

Can be shorter:

```
for ( Loop Ds = loop( "K K pi", "D_s+", FitVertex ) ; Ds ; ++Ds ) {
```

Or:

```
DsForBs2DsPi.DecayDescriptor = "[D_s+ -> K+ K- pi+]cc"
```

GENERIC SELECTION FRAMEWORK

Any (flavour-) physics analysis is a sequence of $A \rightarrow B C (\dots)$, with some cuts in between.

Hence: An analysis algorithm should know:

- Where to get the particles
- What decay to reconstruct
- What cuts to apply
- Where to put the data

Hard-coding cuts is a bad idea . . .

Better to use options of the algorithm

. . . or predefined filters configurable by options:

```
DsForBs2DsPi.MotherCut = "PT > 2000*MeV"
```

Anything else?

DEMO!

```
from math import sqrt
from LoKiPhys.decorators import *
from LoKiCore.functions import monitor
p = LHCb.Particle()
p.setParticleID( LHCb.ParticleID(11) )
m = p.momentum()
m.SetPx ( 1000 )
m.SetPy ( -1000 )
m.SetPz ( 10000 )
m.SetE ( sqrt( m.P2() + 5000*5000 ) )
p.setMomentum ( m )
fun = PX+PY
print PX(p), PY(p) , fun(p)
fun2 = PX>750
print fun2(p)
fun3 = monitor(fun2)
print fun3(p)
from LoKiCore.doxygenurl import browse
browse(PT)
```

Stolen from Vanya

CombineParticles

CombineParticles does exactly that: it combines particles into other particles. It has 5 steps:

- 1 Collect and filter input particles
- 2 Loop over daughters
- 3 Filter the combination
- 4 Fit the vertices and make the mothers particle
- 5 Filter the mothers

Syntax :

```
from Configurables import CombineParticles
myPhi = CombineParticles("MyPhi")
myPhi.InputLocations = [ "StdLooseKaons" ]
myPhi.DecayDescriptor = "phi(1020) -> K+ K-"
```

1. THE DECAY DESCRIPTOR

You need to declare the decay to build. Valid are

```
Jpsi2MuMu.DecayDescriptor = "J/psi(1S) -> mu+ mu-"
HltSharedKstar2KPi.DecayDescriptor = "[K*(892)0 -> K+ pi-]cc"
HltSharedDs2KKPi.DecayDescriptor = "[D_s- -> K+ K- pi-]cc"
# cc needed or D* gets confused
HltSharedD02KK.DecayDescriptor = "[D0 -> K+ K-]cc"
# Cabibbo-favoured and double-suppressed
HltSharedDstarWithD02KPi.DecayDescriptors = [ "[D*(2010)+ -> pi+ D0]cc",
                                                "[D*(2010)+ -> pi+ D~0]cc"
```

The DecayDescriptor does not use the MC truth decay descriptor, but understands only simple things. Don't try

```
[B_s0 -> (J/psi(1S) -> mu+ mu- {,gamma} {,gamma})
      (eta -> pi+ pi- ( pi0 -> gamma gamma))]cc
```

2. CUTS APPLIED ON DAUGHTERS

DaughtersCuts allows you to apply cuts on the incoming daughters.

- To require that the muons have $p_T > 1 \text{ GeV}$ do

```
Jpsi2MuMu.DaughtersCuts = { "mu+" : "(PT>1*GeV)" }
```

It will be applied to both μ^+ and μ^- .

- To also require that the muons have $IP > 2 \sigma$ do

```
Jpsi2MuMu.DaughtersCuts = { "mu+" : '(PT>1*GeV) &  
                                (MIPCHI2DV(PRIMARY)>4)' }
```

...i.e. The χ^2 increase of any primary vertex when adding this muon is more than 4.

- If you have several PIDs involved:

```
HltSharedD02KPi.DaughtersCuts = { "K+" : '(PT>300*MeV) & (P>2*GeV) &  
                                        (MIPCHI2DV(PRIMARY)>4)',  
                                   "pi+" : '(PT>500*MeV) &  
                                             (MIPCHI2DV(PRIMARY)>4)'} }
```


3. CUTS APPLIED ON COMBINATION

CombinationCut allows you to apply cuts on the combination of particles **before** the vertex fit is done.

- To require that the mass is within 30 MeV of the J/ψ

```
jpsi2mumu.CombinationCut = "ADAMASS('J/psi(1S)')<30"
```

- You cut also cut on the p_T of the J/ψ

```
jpsi2mumu.CombinationCut = "(ADAMASS('J/psi(1S)')<30) & (APT>1*GeV)"
```

WARNING: These cuts are applied to the combination (array) of daughters.

- You can only apply cuts which make sense in this case
- They have a different name (by an A)
- The mass of long-lived particles may be different before and after the fit (e.g. K_S^0) → Apply a mass cut after the fit as well.

4. CUTS APPLIED ON MOTHER

CombineParticles applies DVAlgorithm 's default fitter. MotherCut allows you to apply cuts on the **after** the vertex fit is done.

- To require that the Vertex fit has a good χ^2

```
Jpsi2MuMu.MotherCut = "(VFASPF(VCHI2/VDOF)<100)"
```

where VFASPF allows to access the Vertex from the Particle .

That was all we needed

See <https://twiki.cern.ch/twiki/bin/view/LHCb/LoKiHybridFilters>

NEED TO FILTER WITHOUT MAKING?

Sometimes it is helpful to filter what is on the TES:

```
psifilter = FilterDesktop("psifilter")
psifilter.InputLocations = [ "Jpsi2MuMu" ]
psifilter.Code = '(PT>1*GeV) &
                  (MINTREE(ABSID==\'mu+\',P)>8*GeV)'

tutorialseq.Members += [ psifilter ]
```

→ Cuts on the momentum of the muons and on the p_T of the mother.

PLOT TOOL

CombineParticles and FilterDesktop have an integrated PlotTool.

```
from Configurables import LoKi__Hybrid__PlotTool as PlotTool
import GaudiKernel.SystemOfUnits as Units
bs2jpsiphi.HistoProduce = True
bs2jpsiphi.addTool( PlotTool("DaughtersPlots") )
bs2jpsiphi.DaughtersPlots.Histos = { "P/1000" : ('momentum',0,100) ,
                                     "PT/1000" : ('pt_%1%',0,10,200) ,
                                     "M" : ('mass in MeV_%1%_%2%_%3%',0.8*Units.GeV,4*Units.GeV) }
bs2jpsiphi.addTool( PlotTool("MotherPlots") )
bs2jpsiphi.MotherPlots.Histos = { "P/1000" : ('momentum',0,100) ,
                                   "PT/1000" : ('pt_%1%',0,10,200) ,
                                   "M" : ('mass_%1%_%2%_%3%',4*Units.GeV,6*Units.GeV) }
```

The syntax is the same as for the cuts.

PLOT TOOL

