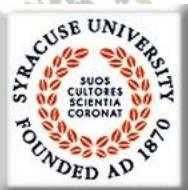




Bender v7r0 as your analysis environment

Vanya BELYAEV



References

LHcb
~~FNAL~~

- Bender Pages and Bender pages by Lena Mayatskaya
- Bender mailing list
- Bender Savannah portal (news, bugs, tasks, ...)
- Bender Tutorial: slides & instructions
- Bender HyperNews, TWiki, FAQ, User Guide and Manual : ☺ not yet. still in the bottle of inc
- Bender Examples
 - including nice scripts from Diego for $B_s \rightarrow \mu\mu$ background studies
getpack Ex/BenderExample v7r0
- “**Bender-helpdesk@lhcb.cern.ch**”
 - 1-R-010 at CERN
 - +41 (0) 22 767 89 28



When use Bender

LHCb
~~FHCb~~

- **Python:** perfect for prototyping
 - e.g. develop the cuts for preselection/stripping
- *Interactive:* perfect for “short” (“supervising”) tasks
 - resolutions
 - spectra
 - “reflections”
- *Flexible & Friendly:*
 - good for “the final” analysis of small data sets
 - combine with **Root**, **Panoramix**, **RooFit**,...



When no Bender



- Stripping does not support **Bender**.
- Reasons?
 - ☹ *Some CPU penalty* for **Bender** selections vs **LoKi** selections is unavoidable (**Python** vs **C++**)
 - could be visible/sizeable for “minimal” job
 - mainly comes from the explicit loops, ntuples and explicit manipulations with dictionaries:
`sqrt(p.px() * p.px() + p.py() * p.py())`
 - could be very small for realistic selection
 - And of course for well-coded lines

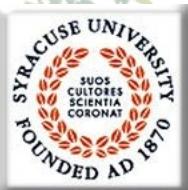
Negligible with patterns (C++) ☺



Bender v7r0

LHCb
~~FHCP~~

- The most fresh version of **Bender**, based on **DaVinci v19r1** - official DC06 stripping version
- The tutorial slides are attached to the agenda
 - Here only some highlights:
 - It is already slide #5, and I have only 30 minutes
 - *If somebody needs*, I would be happy to organize "*hands-on*" **Bender tutorial** similar to tutorials in Beijin & Dortmund or semiprivate tutorial for **HLT** guys.



Minimal Bender script



```
from bendermodule import *
```

Well, It is not
Bender, it is
GaudiPython

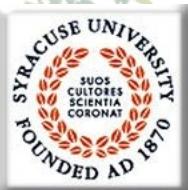
```
gaudi.config( files =  
    [ 'MyOptionsFile.opt' ] )
```

```
gaudi.run(10)
```

Take care about input data!!

```
gaudi.exit()
```

[..../solution/Minimalistic_0.py](#)



Minimal Bender module



```
from bendermodule import *
```

Application and Components Configuration

```
def configure() :  
    gaudi.config( files =  
                  [ 'MyOptionsFile.opts' ] )  
    return SUCCESS  
  
if __name__ == '__main__' :  
    configure()  
    gaudi.run(100)
```

Job steering

[.../solutions/Minimalistic.py](#)



Scripts vs modules



- Dilemma in Python: scripts vs modules
- Scripts are a bit more intuitive and a bit easier to write
 - Problems with reusing ☺
- Modules require some discipline & conventions ☹
 - full power of OO, including classes & extensions
 - Easy to import and reuse ☺
 - the only way to assemble "large" application from pieces
- Be friendly: code modules
 - loose nothing
 - gain a lot



Scripts versus modules



- Script above:

```
import myscript
```

Will execute everything out of control

- Module above:

```
import mymodule  
mymodule.config()  
gaudi.run(100)
```



“Hello, World!” (I)



- The simplest possible BENDER “algorithm”
- Follow LoKi’s style:
 - *inherit the algorithm from useful base class*
 - (re)implement the “analyse” method

```
class HelloWorld(Algo) :  
    def analyse( self ) :  
        print 'Hello, World!'  
        return SUCCESS
```

.../solutions/HelloWorld.py



“Hello, World!” (II)



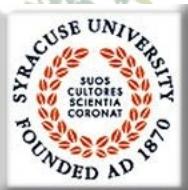
- One needs to instantiate the algorithm
`alg = HelloWorld('Hello')`
- Add it to the list of 'active' algorithms
`gaudi.addAlgorithm(alg)`

Application Configuration

- Execute ☺
`gaudi.run(10)`

Part of job steering block

`..../solutions>HelloWorld.py`



Access to the data (LoKi's style)



- C++: GaudiAlgorithm/LoKi

```
const MCParticles* mcps =  
get<MCParticles>( 'MC/Particles' )
```

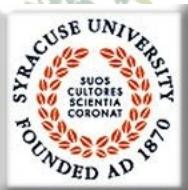
Semantics to be improved

- Python: Bender

- Get as 'native' object:

```
mcps = self.get( 'MC/Particles' )
```

[.../solutions/DataAccess.py](#)



Access to the data using service



- Inside the algorithm

```
dataSvc = self.evtSvc()  
hdr      = dataSvc['Header']  
print 'Event #', hdr.evtNum()
```

No gain

- Outside the algorithms

```
dataSvc = gaudi.evtSvc()  
hdr      = dataSvc['Header']  
print 'Run #', hdr.runNum()
```

The only way!



Attributes and (python) loops



```
for mcp in mcps :  
    print 'ID=' , nameFromPID( mcp.particleID() )  
    print 'PX=' , mcp.momentum().px()  
    print 'PY=' , mcp.momentum().py()
```

MCParticle

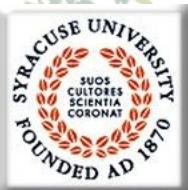
- To know the available attributes:

```
help( obj )  
help( type( obj ) )  
dir(gbl)
```

From Dictionaries

- ON-LINE help for ALL Python/Bender functions/classes. sometimes it is VERY useful

.../solutions/DataAccess.py



Lets start with physics analysis



- >95% of LoKi's idioms are in Bender
- The semantic is VERY similar
 - In spite of different languages
 - few 'obvious' exceptions
- In the game:
 - All Functions/Cuts
 - a bit more round braces are required
 - All (v,mc,mcv) select methods
 - loops , plots
 - for N-Tuples the functionality is a bit limited
 - A lack of template methods,
 - 'farray' need to be validated

Pere knows solution!

Start from MC-truth (requires no special configurations)



MCselect statement



- Selection of **MCParticles** which satisfy the certain criteria:

LUG, Tab. 13.4, p.84

```
mcmu = self.mcselect( 'mcmu' ,  
                      'mu+' == MCABSID )
```

Select μ^+ & μ^-

```
beauty = self.mcselect('beauty' , BEAUTY )
```

```
muFromB = self.mcselect ( 'muFromC' ,  
                           mcmu ,  
                           FROMMCTREE( beauty ) )
```

Everything which has b or b̄

```
muPT = self.mcselect( 'withPT' ,  
                       muFromB ,  
                       ( MCPT > 1000 ) )
```

Everything from
“decay” trees
(incl. decay-on-flight)

.../solutions/MCMuons.py



Change input data



- Get and configure EventSelector

```
evtSel = gaudi.evtSel()
```

```
evtSel.open( "file" )
```

OR

```
evtSel.open( [ "file1", "file2" ] )
```

List of input files

- e.g.

```
evtSel.open ( 'LFN:/lhcb/production/DC04/v1/DST/00000543_00000017_5.dst' )
```



Find MC-tree (IMCDecayFinder)



Brilliant tool from O.Dormond

- find the MC-decay trees:

```
mc = self.mcFinder()
```

```
trees = mc.find(
```

```
  '[B_s0 -> (J/psi(1S) -> mu+ mu-) phi(1020)]cc' )
```

Container("Range") of
MCParticles

- find MC-decay tree components:

```
phis = mc.find(
```

```
' phi(1020) : [B_s0 -> (J/psi(1S) -> mu+ mu-) phi(1020)]cc' )
```

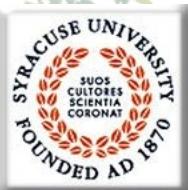
Container("Range") of
MCParticles

- extract 'marked' MC-decay tree components:

```
mus = mc.find(
```

```
  '[B_s0 -> (J/psi(1S) -> mu+ ^mu-) phi(1020)]cc' )
```

.../solutions/MCTrees.py



Add simple histos!

```
for mu in mus :  
    self.plot ( MCPT( mu ) / 1000 ,  
                'PT of muon from J/psi' ,  
                0 , 10 )
```

MCParticle

The default values : #bins = 100, weight = 1

- Configuration for HBOOK histograms:

To be improved!

```
gaudi.HistogramPersistency = 'HBOOK'  
hsvc = gaudi.service('HistogramPersistencySvc')  
hsvc.OutputFile = 'myhistos.hbook'
```

.../solutions/MCTrees.py



Add the simple N-Tuple



```
tup      = self.nTuple( 'My N-Tuple' )
zOrig = MCVXFUN( MCVZ )
for mu in mus :
    tup.column( 'PT' , MCPT   ( mu )   )
    tup.column( 'P'  , MCP    ( mu )   )
    tup.column( 'Z'  , zOrig  ( mu )   )
tup.write()
```

- Configuration:

```
myAlg = g.algorithm( 'McTree' )
myAlg.NTupleLUN = 'MC'
ntsvc = g.service('NTupleSvc')
ntsvc.Output =
[ "MC DATAFILE='tuples.hbook' TYP='HBOOK' OPT='NEW' " ]
```

To be improved

[..../solutions/MCTrees.py](#)



Component Properties



- Algorithms

```
alg = gaudi.algorithm('MyAlg')
alg.NTupleLUN = 'LUNIT'
```

```
MyAlg.NTupleLUN = "LUNIT" ;
```

- Services

```
HistogramPersistencySvc.OutputFile = "histo.file";
```

```
hsvc = gaudi.service('HistogramPersistencySvc')
hsvc.OutputFile = 'histo.file'
```

- Tools

```
MyAlg.PhysDesktop.InputLocations = {"Phys/stdLooseKaons"};
```

```
tool = gaudi.property('MyAlg.PhysDesktop')
tool.InputLocations = ['Phys/StdLooseKaons']
```



./solutions/MCTrees.py



```
# The algorithmm itself
class MCTrees( AlgoMC ) :
    """ The algorithmm itself """

    ## the main analysis method
    def analyse( self ) :
        """ the main analysis method """

        ## get the MCDecayFinder wrapper
        finder = self.mcFinder()

        ## find all MC trees of interest
        trees = finder.find(
            ' [B_s0 -> ( Jpsi(1S) -> mu+ mu- ) phi(1020) ]cc' )

        ## get all kaons from the tree :
        phis = finder.find(
            ' [B_s0 -> ( Jpsi(1S) -> mu+ mu- ) ^phi(1020) ]cc' )

        ## get marked particles from the tree:
        mus = finder.find(
            ' [B_s0 -> ( Jpsi(1S) -> ^mu+ ^mu- ) phi(1020) ]cc' )

        print ' found MCtrees/Phis/Mus: %s/%s/%s' % ( trees.size() ,
                                                       phis.size() ,
                                                       mus.size() )

        ## fill the histogram
        for mu in mus :
            self.plot( MCPT(mu) / 1000 ,
                      ' PT of Muons from J/psi ' ,
                      0 , 10 )

        ## retrieve (bon-on-demand) N-Tuple
        tup = self.nTuple( 'My N-Tuple' )
        zOrig = MCVXFUN( MCVZ )

        for mu in mus :
            tup.column( 'P' , MCP( mu ) / 1000 )
            tup.column( 'PT' , MCPT( mu ) / 1000 )
            tup.column( 'ID' , MCID( mu ) )
            tup.column( 'Q3' , MC3Q( mu ) )
            tup.column( 'ZOR' , zOrig( mu ) )
            tup.write()

        return SUCCESS
# =====
```

```
# =====
## configure the job
def configure() :
    """ configure the job """

    gaudi.config( files = ['$DAVINCIROOT/options/DaVinciCommon.opts'] )

    # 1) create the algorithm
    alg = MCTrees( 'McTree' )

    # 2) replace the list of top level algorithm by only *THIS* algorithm
    gaudi.setAlgorithms( [ alg ] )

    if 'HbookCnv' not in gaudi.DLLs : gaudi.DLLs += ['HbookCnv']
    gaudi.HistogramPersistency = "HBOOK"
    hps = gaudi.service('HistogramPersistencySvc')
    hps.OutputFile = 'MTrees_histos.hbook'

    # add the printout of the histograms
    hsvc = gaudi.service( 'HbookHistSvc' )
    hsvc.PrintHistos = True

    # configure the N-Tuples:
    ntsvc = gaudi.ntuplesvc()
    ntsvc.Output = [ "MC DATAFILE='MCTrees.hbook' OPT='NEW' TYP='HBOOK'" ]

    # configure my own algorithm
    myAlg = gaudi.algorithm('McTree')
    myAlg.NTupleLUN = 'MC'
    myAlg.PP2MCs = []

    ## redefine input files
    evtSel = gaudi.evtSel()
    evtSel.PrintFreq = 50
    import data tutorial as data
    evtSel.open( data.FILES )

    return SUCCESS
# =====
## Job steering
if __name__ == '__main__':
    ## job configuration
    configure()
    ## event loop
    gaudi.run(100)
# =====
```



Go from MC to RC data



- At this moment one knows how to:
 - Deal with MC trees, decays, particles
 - Perform simple (**python**) loops
 - Deal with the histograms & N-Tuples
 - Some knowledge of 'configuration'
- For RC data one must perform non-trivial algorithm configuration to be able to run
 - Input for RC particles (or **ParticleMaker**)
 - Dependency on 'other' algorithms ('**PreLoad**')



Algorithm configuration

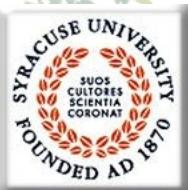


```
desktop = gaudi.property('MyAlg.PhysDesktop')
desktop.InputLocations = [ "Phys/StdLooseKaons" ]
```

- Similar semantic in configuration ('*' .opts) files:

```
MyAlg.PhysDesktop.InputLocations={ "Phys/StdLooseKaons" } ;
```

.../solutions/RCSelect.py



select/loop statements



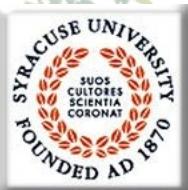
LUG, Tab. 13.2, p.62-77

```
muons = self.select( 'mu' ,  
                      ( 'mu+'== ABSID ) & ( PT > (1*GeV) ) )  
  
kaons = self.select( 'K' ,  
                      ( 'K+'== ABSID ) & ( PIDK > 0 ) )
```

- Loops:

```
psis=self.loop( 'mu mu' , 'J/psi(1S)' )  
phis=self.loop( 'K K' , 'phi(1020)' )
```

[.../solutions/RCSelect.py](#)



Inside the loops (I)

```
dmcut = ADMASS('J/psi(1S)') < 50
for psi in psis :
    if not 2500 < psi.mass(1,2) < 3500 : continue
    if not 0 == SUMQ( psi ) : continue
    if not 0 <= VCHI2( psi ) < 49 : continue
    self.plot ( M(psi)/1000 ,
                " di-muon invariant mass" ,
                2.5 , 3.5 )
    if not dmcut( psi ) : continue
    psi.save('psi')

psis = self.selected('psi')
print '# of selected J/psi candidates:', psis.size()
```

.../solutions/RCSelect.py



Inside the loops (II)



```
dmcut = ADMASS('phi(1020') < 12
for phi in phis :
    if not phi.mass(1,2) < 1050      : continue
    if not 0 == SUMQ( phi )          : continue
    if not 0 <= VCHI2( phi ) < 49 : continue
    self.plot ( M( phi ) / 1000 ,
                " di-kaon invariant mass" ,
                1.0 , 1.050 )
    if not dmcut( phi ) : continue
    phi.save('phi')
```

$\Sigma q = 0$

$\chi^2_{VX} < 49$

$|\Delta M| < 12 \text{ MeV}/c^2$

```
phis = self.selected('phi')
print '# of selected phi candidates:', phis.size()
```

[.../solutions/RCSelect.py](#)



Inside the loops (III)



```
dmcut = ADMASS('B_s0') < 100
bs = self.loop( 'psi phi' , 'B_s0' )
for B in bs :
    if not 4500 < B.mass(1,2) < 6500 : continue
    if not 0 <= VCHI2( B ) < 49 : continue
    self.plot( M( B ) / GeV ,
               " J/psi phi invariant mass" ,
               5.0 , 6.0 )
    if not dmcut( B ) : continue
    B.save('Bs')

Bs = self.selected('Bs')
print '# of selected Bs candidates:', Bs.size()
if not Bs.empty() : self.setFilterPassed( TRUE )
```

[.../solutions/RCSelect.py](#)



The last step: MC-truth match



- The simplest case: check if RC particle originates from the certain MC-(sub)tree
 - The most frequent case
 - Check for efficiencies
 - Resolution
- The opposite task: what MC particle “corresponds” to RC particle
 - similar ($\text{MCTRUTH} \rightarrow \text{RCTRUTH}$)
- NB: **LoKi** (and **Bender**) uses own concept of MC “loose” matching
 - LUG, chapter 15



MC-truth match



```
finder = self.mcTruth('some name')
```

• Select MC-particles

```
mcBs = finder.find(  
    ' [B_s0 -> (J/psi(1S) -> mu+ mu-) phi(1020)]cc ' )
```

```
mcPhi = finder.find(  
    ' phi(1020) : [B_s0 -> (J/psi(1S) -> mu+ mu-) phi(1020)]cc ' )
```

```
mcPsi = finder.find(  
    ' J/psi(1S) : [B_s0 -> (J/psi(1S) -> mu+ mu-) phi(1020)]cc ' )
```

• Prepare 'MC-Truth cuts'

```
match = self.mcTruth('some name')  
mcCutBs = MCTRUTH ( match , mcBs )  
mcCutPhi = MCTRUTH ( match , mcPhi )  
mcCutPsi = MCTRUTH ( match , mcPsi )
```

[..../solutions/RCMCSelect.py](#)



The last step: MC-truth match



```
for psi in psis :  
    if not mccutPsi ( psi ) : continue  
    ...  
for phi in phis :  
    if not mccutPhi ( phi ) : continue  
    ...  
for B in bs :  
    if not mccutBs ( B ) : continue  
    ...
```

}

.../solutions/RCMCSelect.py

• Alternatively :

```
for B in bs :  
    psi = B(1)  
    phi = B(2)  
    ...  
    tup.column ( 'mcpsi' , mccutPsi( psi ) )  
    tup.column ( 'mcphi' , mccutPhi( phi ) )  
    tup.column ( 'mc' , mccutBs ( B ) )  
    tup.write()
```



.../solutions/RCMCSelect.py



```

# =====
author__ = 'Vanya BELYAEV ibelyaev@physics.syr.edu'
# =====
## import everything from BENDER
from bendermodule import *
# =====
## @class RCMCSelect
## my analysis algorithm
class RCMCSelect(AlgoMC):
    """ my analysis algorithm """
    ## the main analysis method
    def analyse( self ) :
        """ the main analysis method """
        ## get MCDecayFinder wrapper:
        finder = self.mcFinder()
        ## find all MC trees
        mcBs = finder.find(
            '[ B_s0 -> ( J/psi(1S) -> mu+ mu- ) phi(1020)]cc' )
        ## get all MC phis from the tree :
        mcPhi = finder.find(
            '[ B_s0 -> ( J/psi(1S) -> mu+ mu- ) ^phi(1020)]cc' )
        ## get all MC psis from the tree :
        mcPsi = finder.find(
            '[ B_s0 -> ( ^J/psi(1S) -> mu+ mu- ) phi(1020)]cc' )
        ## get helper object for MC-match
        match = self.mcTruth( 'MCdecayMatch' )
        ## prepare Monte-Carlo Cuts"
        mcCutBs = MCTRUTH( match , mcBs )
        mcCutPhi = MCTRUTH( match , mcPhi )
        mcCutPsi = MCTRUTH( match , mcPsi )
        ## select muons for J/Psi reconstruction
        muons = self.select( "mu" , ( "mu+" == ABSID ) & ( PT > 500 ) )
        if muons.empty() : return SUCCESS
        ## select kaons for Phi reconstruction
        kaons = self.select( "K" , ( "K+" == ABSID ) & ( PIDK > 0.0 ) )
        if kaons.empty() : return SUCCESS
        ## delta mass cut for J/psi
        dmPsi = ADMASS('J/psi(1S)') < 50
        ## prepare the loop over dimuons
        psis = self.loop( 'mu mu' , 'J/psi(1S)' )
        for psi in psis :
            ## use *ONLY* Monte-Carlo cuts
            if not mcCutPsi( psi ) : continue      ## ATTENTION! only true J/psi
            ## rough estimation of the mass
            mass = psi.mass(1,2) / 1000
            if not 2.5 < mass < 3.5 : continue
# \*** RCMCSelect.py          (Pythron ADVANCE CVS:1.9)--L16--CO--10%

```

```

## neutral combination?
if not 0 == SUMQ( psi ) : continue
## check the chi2 of the vertex fit
if not 0 <= VCHI2( psi ) < 49 : continue
self.plot( M(psi) / 1000 ,
           " dimuon invariant mass " ,
           2.5 , 3.5 )
if not dmPsi( psi ) : continue
psi.save( 'psi' )                                ## save J/psi
## delta mass cut for phi
dmPhi = ADMASS('phi(1020)') < 20
## prepare the loop over dikaons
phis = self.loop( 'K K' , 'phi(1020)' )
for phi in phis :
    ## use *ONLY* Monte-Carlo cuts
    if not mcCutPhi( phi ) : continue ## ATTENTION: only true phi
    if phi.mass( 1 , 2 ) > 1050 : continue
    # neutral combination ?
    if not 0 == SUMQ( phi ) : continue
    if not 0 <= VCHI2( phi ) < 49 : continue
    self.plot( M(phi) / 1000 ,
               " dikaon invariant mass " ,
               1.0 , 1.050 )
    if not dmPhi( phi ) : continue
    phi.save('phi')                                ## save phi
## delta mass cut for Bs
dmBs = ADMASS('B_s0') < 100
## prepare the loop over psi+phi combinations
bs = self.loop( 'psi phi' , 'B_s0' )
for B in bs :
    ## use *ONLY* Monte-Carlo cuts
    if not mcCutBs( B ) : continue ## ATTENTION: only true Bs
    #
    m = B.mass(1,2) / 1000
    if not 4.5 < m < 6.5 : continue
    if not 0 < VCHI2( B ) < 49 : continue
    self.plot( M(B) / 1000 ,
               " psi phi invariant mass " ,
               5.0 , 6.0 )
    if not dmBs( B ) : continue
    B.save('Bs')                                    ## save Bs
    # check selected particles:
    Bs = self.selected('Bs')
    if not Bs.empty() : self.setFilterPassed( True ) ## FILTER PASSED
# =====
return SUCCESS
# =====

```



.../solutions/RCMCSelect.py



```
## Job configuration:
def configure() :
    """ Job configuration """
    gaudi.config ( files = [
        '$DAVINCIROOT/options/DaVinciCommon.opts',
        '$COMMONPARTICLESROOT/options/StandardKaons.opts',
        '$COMMONPARTICLESROOT/options/StandardMuons.opts'
    ] )
## modify/update the configuration:
# 1) create the algorithm
alg = RCSelect( 'RCSelect' )
# 2) add the algorithm
gaudi.addAlgorithm( alg )
# 3) configure algorithm
desktop = gaudi.tool('RCSelect.PhysDesktop')
desktop.InputLocations = [ 'Phys/StdLooseKaons' , 'Phys/StdLooseMuons' ]
## configure the histograms:
if 'HbookCnv' not in gaudi.DLLs : gaudi.DLLs += [ 'HbookCnv' ]
gaudi.HistogramPersistency = "HBOOK"
hps = gaudi.service('HistogramPersistencySvc')
hps.OutputFile = 'RCMCselect_histos.hbook'
## configure the N-Tuples:
ntsvc = gaudi.ntuplervc()
ntsvc.Output = [ "RCMC DATAFILE='HandsOn3.hbook' OPT='NEW' TYP='HBOOK'" ]
## add the printout of the histograms
hsvc = gaudi.service('HbookHistSvc')
hsvc.PrintHistos = True
## configure the desktop:
myAlg = gaudi.algorithm('RCSelect')
myAlg.PP2MCs = [ 'Relations/Rec/ProtoP/Charged' ]
myAlg.NTupleLUN = 'RCMC'
## define the proper input data:
evtSel = gaudi.evtSel()
evtSel.PrintFreq = 20
import data_tutorial as data
evtSel.open( data.FILES )

    return SUCCESS
#
# =====
## Job steering:
if __name__ == '__main__':
    ## job configuration
    configure()
    ## event loop
    gaudi.run(1000)

#
# =====
\*** RCMCSelect.py (Python IDLE/NC CVS-1.9) --L106--CO-723
```

- Algorithm: 81 lines
 - 55% - comments
- Configuration&steering: 44 lines
 - 40% comments
- Select true “reconstructed” Bs with loose cuts: fine for cuts investigation



Other features, out of scope



- Nice printout of trees, particles, events
- Various “extractors” and metafunctions
- HepMC + HepMCParticleMaker
- Jets, Jets maker, LoKi-kt-Jet
- Tools for background origin studies
- Patterns
- “Hybrid”: now also for MCParticles
 - “IFilterCriterion” in python
 - “IMCParticleSelector” in python
- and much much more...

As concerns the functionality needed for analysis, Bender is full scale application, widely used for physics studies



References again...



- Bender Pages and Bender pages by Lena Mayatskaya
- Bender mailing list
- Bender Savannah portal (news, bugs, tasks, ...)
- Bender Tutorial: slides & instructions
- Bender HyperNews, TWiki, FAQ, User Guide and Manual : ☺ not yet. still in the bottle of inc
- Bender Examples
 - including nice scripts from Diego Martitez Santos for $B_s \rightarrow \mu\mu$ background studies
`getpack Ex/BenderExample v7r0`
- “**Bender-helpdesk@lhcb.cern.ch**”
 - 1-R-010 at CERN
 - +41 (0) 22 767 89 28