Bender "Tutorial" v7r0

Vanya BELYAEV

Outline

- **Bender/Python** overview
- Job configuration
- Data access
- Histograms & N-Tuples
- Algorithms

Significant improvements in **Bender** semantics are expected (mainly according to the feedback from you)

Bender is not frozen!

Environment (I)

- **Bender v7r0**
  - The lastest DC06 release
  - based on **DaVinci v19r1,Phys v6r2**
- The package **Tutorial/BenderTutor v7r0**
- Only few essential features of **Bender**
- Out of Tutorial scope
  - visualization of histograms, Panoramix, Root, etc..
  - visualization of event and detector data
  - **CMT**-free mode
  - batch jobs
  - **Bender&GRID**
    - **Bender&DIRAC** by Ying Ying Li
    - **Bender&GANGA** by Karol Hennesy

Environment (II)

- get the Tutorial package

```
setenv Bender v7r0
cd $HOME/cmtuser/Bender_v7r0
getpack Tutorial/BenderTutor v7r0
cd Tutorial/BenderTutor/v7r0/cmt
make
source setup.csh
```

Bender/Python tips

- Python scripts could be executed as "scripts"

```
>    python MyBenderScript.py
>           MyBenderScript.py
```

```
#!/usr/bin/env python2.4
```

- Python scripts could be executed from the command prompt (explicit interactivity!)

```
> python
    >>>     import MyBenderScript
```

- Python scripts could be executed with the command prompt (interactivity like "**pawlogon.kumac**")

```
> python –i MyBenderScript.py
```

Common start-up script is possible,
Pere has a lot of nice ideas!

Structure of Gaudi Job

Each "Job" contains 4 essential part

- Configuration of Job environment
  - <ProjectEnv> scripts, CMT
- Configuration of Job's components
  - Top Level algorithms
  - properties of Algorithms/Services/Tools
  - Input/output
- "Analysis Algorithm" coding
- Job steering

Bender: cmt.py

GaudiPython + Bender

Bender

GaudiPython + Bender

2 approaches

Start from pure python prompt

- define everything from Python

Attractive, but not practical

Make a "smooth" transition from `DaVinci/LoKi`

- start with existing configuration
- substitute it element by element

Choice for tutorial

Minimal Analysis Job

- **Bender** could be used with "no **Bender**"
- Execute some "**DaVinci**" configuration
- The actual configuration from '*'**.opts** file

- **DaVinci:**

```
DaVinci MyOptionsFile.opts
```

Minimal Bender script

```
from   bendermodule import *

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

gaudi.run(10)

gaudi.exit()
```

Take care about input data!!

../solution/Minimalistic_0.py

Minimal Bender script

```
from   bendermodule import *


def configure() :
   gaudi.config( files =
                        ['MyOptionsFile.opts'])
   return SUCCESS

if __name__ == '__main__' :
   configure()
   gaudi.run(100)
```

Application and Components Configuration

Job steering

`../solutions/Minimalistic.py`

"Hello, World!" (I)

- The simplest possible "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

No gain

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

- Outside the algorithms

The only way!

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

Store Browse

- Inside algoritm

```
dataSvc = self.evtSvc()
```

- Outside algorithm

```
dataSvc = gaudi.evtSvc()
```

Browse by directory name

```
dataSvc.dir('/Event/Rec')
```

```
mc = dataSvc['MC']
dataSvc.dir(mc)
dataSvc.ls(mc)
```

Browse by directory itself

alias

Attributes and (python) loops

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

MCParticle

From Dictionaries

- To know the available attributes:

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

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

`../solutions/DataAccess.py`

Hands-on (I)

- Simple algorithm which gets **`MCVertices`** from the Gaudi Transient Store and prints number of **`MCVertices`** and some information (e.g. **`x/y/z`**-position) for some of them

Hints:

- The analogous example for **`MCParticles`**:
  - **`../solutions/DataAccess.py`**
- The actual solution is
  - **`../solutions/HandsOn1.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

Start from MC-truth (requires no special configurations)

Pere knows solution!

MCselect statement

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

LUG, Tab. 13.4, p.84

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

Select μ^+ & μ^-

Everything which has b or b̄

- Refine criteria:

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

Hands On (II, II.5)

- Simple algorithm which evaluates the fractions of events which contains of at least B_s or beauty baryons

Hints

- Relevant `MCParticle` functions

  `MCID, MCABSID , BEAUTY , BAR` LUG, Tab. 13.4, p.84-87
- The most trivial "counter" is

```
nBs  = self.counter("nBs")
nBs += number
```

- The analogous algorithm is
  - `../solutions/MCmuons.py`
- The real solution is
  - `../solutions/HandsOn2.py`
  - `../solutions/HandsOn2.5.py`

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 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

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

```
alg = gaudi.algorithm('MyAlg')
alg.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']
```

Hands On (III)

- The algorithm which gets the kaons from the decay $B_s \to J/\psi (\phi \to K^+ K^-)$, fill histo and N-Tuple

Hints

- One need to define input MC files for this decay
  - see `../solutions/MCTrees.py`
- The similar algorithm
  - `../solutions/MCTrees.py`
- The actual solution
  - `../solutions/HandsOn3.py`

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()
```

$\Sigma q = 0$

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

$|\Delta M| < 50$ MeV/c^2

../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')

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

$\Sigma q = 0$

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

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

../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 (**MCTRUTH** → **RCTRUTH**)
- **NB**: **LoKi** (and **Bender**) uses <u>own</u> 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()
```

Hands On (IV)

- Simple algorithm which selects kaons, plot di-kaon invariant mass with and without MC-truth flags with different **`PIDK`** ($= \Delta_{LL}(K\text{-}\pi)$) values (& fill N-Tuple with such information)

Hints

- The relevant functions/cuts
  - **`PIDK, MCTRUTH`**
- The analogous algorithm
  - **`../solutions/RCMCSelect.py`**
- The actual solution
  - **`../solutions/HandsOn4.py`**

Other information

- Bender Pages
  - Bender pages by *Lena Mayatskaya*
- Bender mailing list
- Bender Savannah portal (news, bugs, tasks, …)
- **Bender Hyper News**
  - ☹ no link: to be launched soon
- **Bender User Guide and Manual**
  - ☹ no link: still in the bottle of inc
- Bender Examples
  - including nice scripts from *Diego* for $B_s \to \mu\mu$ background studies

    `getpack Ex/BenderExample v7r0`
- "**Bender-helpdesk@lhcb.cern.ch**"
  - 1-R-010 at CERN
  - **+41 (0) 22 767 89 28**