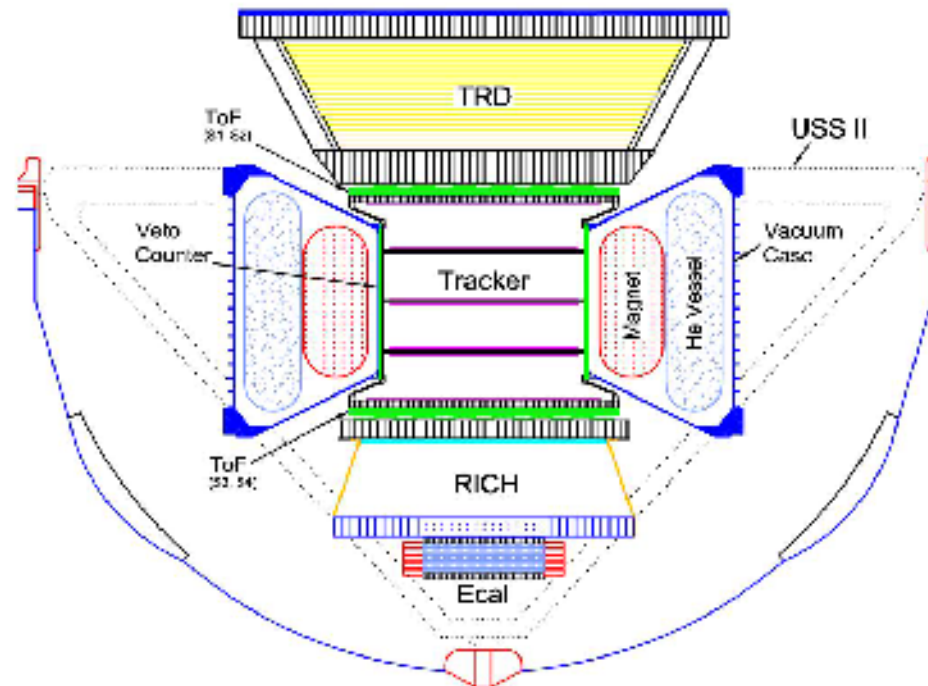


AMS ANALYSIS AND SCRIPTING LANGUAGES



Juan Alcaraz, CIEMAT - Madrid

Splinter Meeting, 22 January 2004

OUTLINE

- Alternatives to C++ ?
- Pro and Cons of scripting languages
- Python and Ruby examples

Alternatives to C++ ?

- Not really, in what concerns a "compiled analysis" (convenient to analyse huge samples).
- In the case of "interpreted" analysis -> scripting languages
- Python and Ruby have been implemented (first versions)

Pro and Cons of scripting languages

■ Cons

- Interpreted languages (although bytecode compiled, like Python), so not ideal for "huge" data samples
- Another language to learn?

■ Pros

- Easy to learn and comfortable to use -> more productivity
- "VERY" High level languages
- Many other modules, libraries and utilities shipped with (Text processing, Cgi/Html, Networking, Graphics, Threads, ...)

Scripting Languages: Very High Level Languages

```
# Example of the possibilities of a Very High Level language: Pyt
# Print all AMS ROOT classes available in any ROOT version
my_file = open("/home/alcaraz/ams/AMS/include/root.h")

for line in my_file:
    match = re.search(r'^class\s+(.+R)\s*[:\{\}]', line)
    if match:
        print "class " + match.group(1)
"python_example.py" 12L, 358C                                12,13                                Bot
```

Now running it...

```
class HeaderR
class EcalHitR
class EcalClusterR
class Ecal2DClusterR
class EcalShowerR
class TofRawClusterR
class TofClusterR
class AntiClusterR
class TrRawClusterR
class TrClusterR
class TrRecHitR
class TrTrackR
class RichHitR
class RichRingR
class TrdRawHitR
class TrdClusterR
class TrdSegmentR
```

Scripting Languages: Very High Level Languages

```
#!/usr/bin/env ruby

# Example of the possibilities of a Very High Level language: Rub
# Print all AMS ROOT classes available in any ROOT version
my_file = open("/home/alcaraz/ams/AMS/include/root.h")

for line in my_file
  if line =~ /^class\s+(.+R)\s*[:\{]/
    puts "class " + $1
  end
end

"ruby_example.rb" 11L, 310C
```

9,13

Top

Now running it...

```
class HeaderR
class EcalHitR
class EcalClusterR
class Ecal2DClusterR
class EcalShowerR
class TofRawClusterR
class TofClusterR
class AntiClusterR
class TrRawClusterR
class TrClusterR
class TrRecHitR
class TrTrackR
class RichHitR
class RichRingR
class TrdRawHitR
class TrdClusterR
class TrdSegmentR
```

Scripting Languages: Comfortable Coding

```
def check_float(input)
  begin
    cut_test = Float(input)
  rescue ArgumentError
    puts "Wrong value of cut \"#{input}\"; ignored"
    nil
  ensure
    puts "This method tests the validity of a Float"
    puts "before returning it"
    input
  end
end

test_this = "<"
check_float test_this
"ruby_comfort.rb" 56L, 1055C                    55,1                    Bot
```

Now running it...

```
We have been inside the loop 1000 times
We have been inside the loop 1000 times
We have been inside the loop 1000 times
We have been inside the loop 1000 times
We have been inside the loop 1000 times
We have been inside the loop 1000 times
Wrong value of cut "<"; ignored
This method tests the validity of a Float
before returning it
```


Ruby

- <http://www.ruby-lang.org/>
- A scripting language created by Yukihiro Matsumoto ("Matz")
- Ruby is more popular than Python in Japan; increasing popularity outside
- Some experts claim that $\text{Ruby} > (\text{Perl} + \text{SmallTalk})/2$
- Principles on Minimum Surprise and "TMTOWTDI"
- Everything in Ruby is an Object, even the number "1"
- Very comfortable when working with classes, blocks, ...

Ruby Language: Example of a Class

```
#!/usr/bin/env ruby
```

```
class Measurement
  attr_accessor :value, :error
  def initialize(a,b)
    @value = a
    @error = b
  end
  def to_s
    "#{value} +- #{error}"
  end
  def + (a)
    value_new = value + a.value
    error_new = Math.sqrt(error**2+a.error**2)
    Measurement.new(value_new,error_new)
  end
end
```

```
"meas.rb" 36L, 843C
```

1,1

Top

Now running it...

```
Measurement A: 8.0 +- 2.82842712474619
Measurement B: 5.0 +- 2.23606797749979
Sum of A and B: 13.0 +- 3.60555127546399
Now A is: 13.0 +- 3.60555127546399
Combination of A and B: 7.22222222222222 +- 1.90029237516523
```

AMS analysis in other languages: `ams_example.py`

```
#!/usr/bin/env python2.2
from AMS import *
app = TApplication("My application", 0, [])
hist = TH1F("hist", "Mass (GeV)", 50, 0.0, 5.0);
ams = AMSChain()
ams.Add("vitaly.root")
ndata = ams.GetEntries()
for i in range(ndata):
    ev = ams.GetEvent()
    if ev.nParticle() == 1:
        hist.Fill(ev.Particle(0).Mass)
hist.Draw()
app.Run()
"ams_example.py" 20L, 338C
```

18,1

A11

Now running it...

AMS analysis in other languages: `ams_example.rb`

```
#!/usr/bin/env ruby

require 'AMS'
include AMS

app = TApplication.new("My application", 0, [] )

hist = TH1F.new "hist", "Mass (GeV)", 50, 0.0, 5.0

ams = AMSChain.new
ams.Add "vitaly.root"
ndata = ams.GetEntries

ndata.times do
  ev = ams.GetEvent
  if ev.nParticle == 1
    hist.Fill ev.Particle(0).Mass
  end
end
```

"ams_example.rb" 23L, 341C

16,3

Top

Now running it...

