

New Machine-Learning Libraries in CVMFS

Pere Mato

IML Meeting, 24th October 2016

Outline

- ❖ Introduction to LCG Releases and Views
- ❖ Use cases and Examples
- ❖ ML packages
- ❖ Management process

LCG Releases

- * Configuring, building and deploying **external libraries** (~300) and **MC Generators** (~80) for all the supported (~12) platforms (3 os, 8 compilers) used by LHC experiments
 - * Releasing full configurations. Content, versions and platforms discussed / agreed with experiments (LIM+AF)
 - * The EP-SFT groups have been providing this service to the experiments successfully for the last 10 years
 - * Recently we have been adding many packages for data analysis (Python, R, Machine Learning)
- * **Implementation**
 - * Builds are done with a home-made tool based on **CMake** (LCGCMake)
 - * The output are RPM / Tarfiles that are installed in AFS and CVMFS and available to experiments
 - * Generation of 'Views'

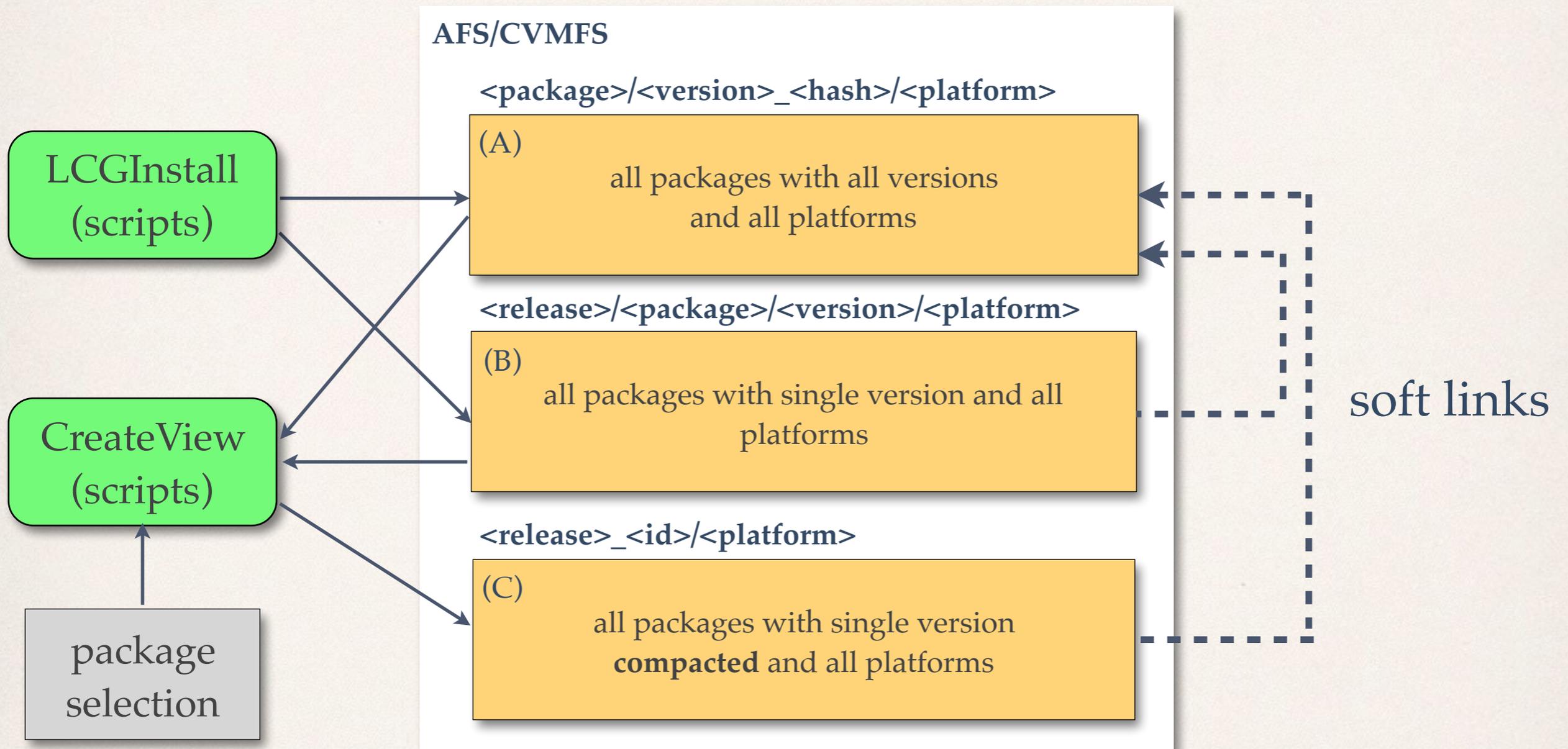
Configurations

- * An LCG configuration is a **named** and **immutable** list of (package, version) tuples
 - * Examples:
 - * LCG_84, LCG_85, LCG_85a
- * We also provide every night a **mutable** release candidate for different purposes: **nightlies**
 - * For the experiments to test new versions, for users to play, etc.
 - * Understanding compatibility and overall integration

pyqt	4.11.4
pyqt5	5.5.1
pytest	2.2.4
pytz	2015.4
Python	2.7.10
PythonFWK	2.7.10
python_dateutil	2.4.0
pytools	1.9
PyYAML	3.11
pyxml	0.8.4p1
pyzmq	14.5.0
QMtest	2.4.1
Qt	4.8.7
Qt5	5.6.0
qwt	6.0.1
R	3.2.0
scikitlearn	0.16.1
scipy	0.15.1

<http://lcgsoft.web.cern.ch>

LCG Views



```
/cvmfs/sft.cern.ch/lcg/views/<release>/<platform>/bin  
lib  
include  
...
```

Budding a View: Selection

- ❖ Defining the set of packages
 - ❖ The default is to take all packages in a LCG release
 - ❖ Sub-setting is possible
- ❖ Only one version per package
 - ❖ Explicit selection for multi-version packages (i.e. MC generators)
 - ❖ Packages with major versions (i.e. Qt4 vs. Qt5, Python2 vs. Python3)
- ❖ Selection of top level directories
 - ❖ Many unneeded files with big chances of clash (README, ...)
 - ❖ Current list: ['aclocal', 'cmake', 'emacs', 'fonts', 'include', 'macros', 'test', 'tests', 'bin', 'config', 'etc', 'icons', 'lib', 'lib64', 'man', 'tutorials', 'share']

Environment Setup

```
$ source /cvmfs/sft.cern.ch/lcg/views/<release>/<platform>/setup.[c]sh
```

- ❖ Sourcing a single and simple file sets the full environment for the complete view of LCG release. It defines trivially:
 - ❖ PATH, LD_LIBRARY_PATH
 - ❖ PYTHONPATH
 - ❖ CMAKE_PREFIX_PATH
 - ❖ ROOTSYS, ROOT_INCLUDE_PATH
- ❖ Other variables can be added if needed...
 - ❖ E.g. SPARK_HOME, JUPYTER_PATH, etc.

Use Cases: Jupyter Notebooks

```
$ source /cvmfs/sft.cern.ch/lcg/views/dev3/latest/x86_64-slc6-gcc49-opt/setup.csh  
$ jupyter notebook
```

- ❖ The command will open a web browser in which the user can create his/her notebooks
- ❖ The notebook will inherit the full environment
 - ❖ All Python modules available
 - ❖ All HEP libraries available
 - ❖ E.g. ROOT, Fastjet, Geant4, ...

CernStaff with Pandas and ROOT

```
In [6]: %matplotlib inline  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import ROOT
```

Welcome to JupyROOT 6.07/02

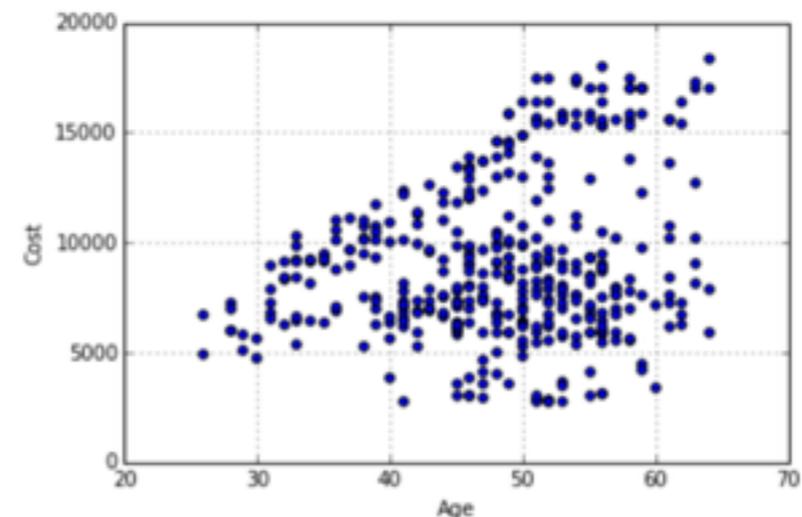
Read the cern staff data in CSV format and display the first few rows.

```
In [7]: cernstaff = pd.read_csv('cernstaff.csv')
```

Produce a scatter plot of Cost versus Age

```
In [10]: (cernstaff[cernstaff.Department == 'EP']  
.plot(kind='scatter', x='Age', y='Cost'))
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc9bf162b10>
```



Use Cases: SWAN

SWAN Customisation

Specify the parameters that will be used to contextualise the container which is created for you. See [the online SWAN guide](#) for more details.

Software stack [more...](#)

- ✓ 85 SWAN3
- 85 SWAN2
- Development Bleeding Edge (might be unstable)

Environment script [more...](#)

Number of cores [more...](#)

[Start my Session](#)

Here a list of stable releases will by made available

Latest successful nightly. Ideal for testing new packages and versions

Use Case: Building Software

```
#-Build and Run the Event ROOT example-----  
wget http://root.cern.ch/download/event.tar.gz  
tar -zxf event.tar.gz  
cd event/build  
cmake ..  
make -j10  
./Run
```

ROOT

```
#-Build B1 basic Geant4 example-----  
mkdir build; cd build  
cmake <lcg-view>/share/Geant4-10.1.2/examples/basic/B1  
make -j10
```

```
#-Run B1 basic Geant4 example (needs data files)-----  
setenv G4DATA /cvmfs/geant4.cern.ch/share/data  
setenv G4NEUTRONHPDATA $G4DATA/G4NDL4.5  
...  
./exampleB1 run1.mac
```

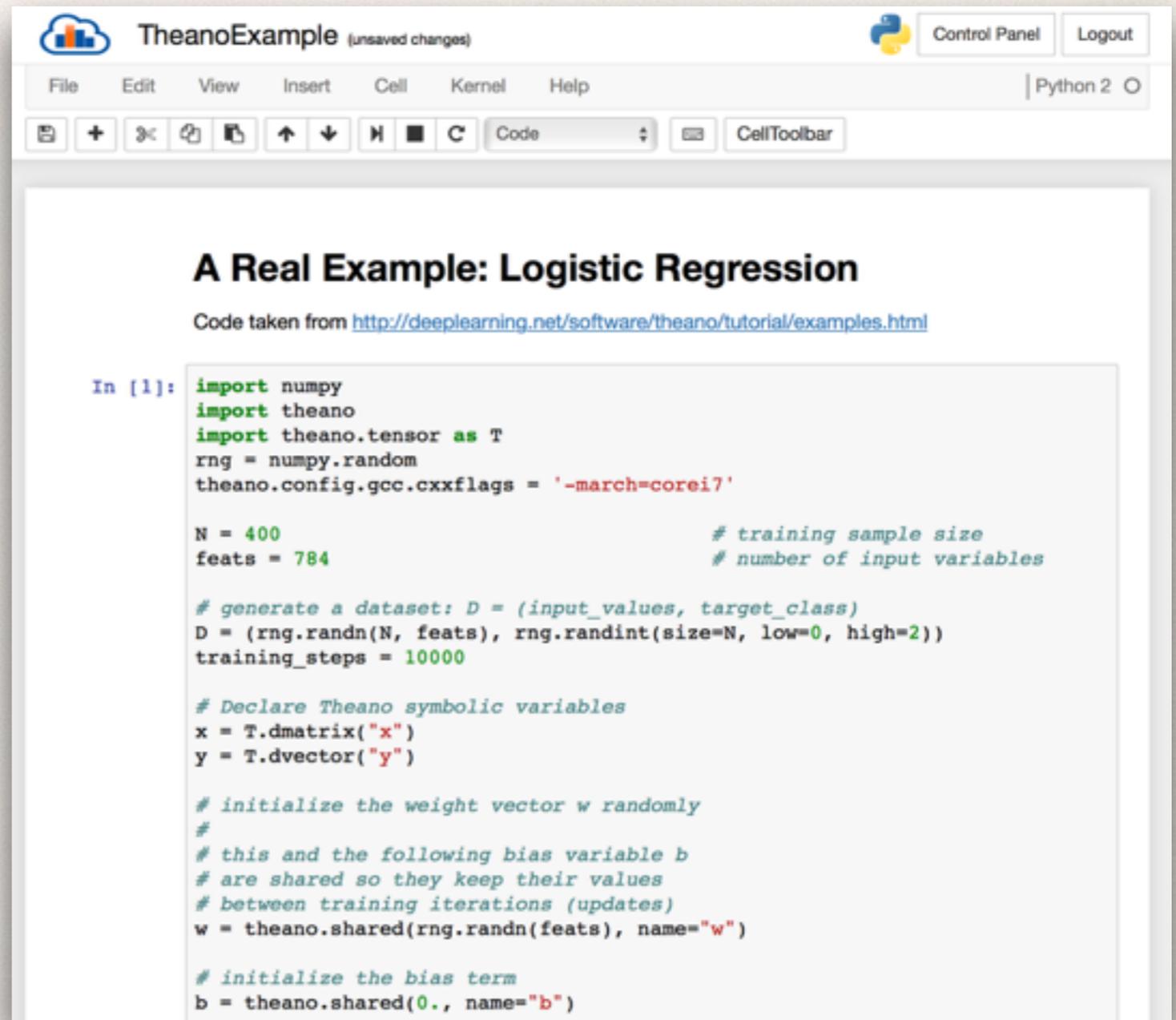
Geant4

Currently Installed ML Packages

- ❖ Python scientific data analysis packages
 - ❖ Numpy, Scipy, Pandas, matplotlib, etc.
- ❖ ROOT and Python
 - ❖ ROOT, rootpy, root_numpy
- ❖ Machine Learning
 - ❖ ROOT/TMVA
 - ❖ ScikitLearn
 - ❖ Theano
 - ❖ Keras

Example: Theano in SWAN

- ❖ Selecting 'Development Bleeding Edge' software stack you can start using Theano
 - ❖ One little problem with the actual hardware processor: the runtime architecture might not be same as the build architecture
- ❖ We are not experts on these tools, so feedback would be highly appreciated



TheanoExample (unsaved changes) Control Panel Logout

File Edit View Insert Cell Kernel Help Python 2

Code CellToolbar

A Real Example: Logistic Regression

Code taken from <http://deeplearning.net/software/theano/tutorial/examples.html>

```
In [1]: import numpy
import theano
import theano.tensor as T
rng = numpy.random
theano.config.gcc.cxxflags = '-march=corei7'

N = 400 # training sample size
feats = 784 # number of input variables

# generate a dataset: D = (input_values, target_class)
D = (rng.randn(N, feats), rng.randint(size=N, low=0, high=2))
training_steps = 10000

# Declare Theano symbolic variables
x = T.dmatrix("x")
y = T.dvector("y")

# initialize the weight vector w randomly
#
# this and the following bias variable b
# are shared so they keep their values
# between training iterations (updates)
w = theano.shared(rng.randn(feats), name="w")

# initialize the bias term
b = theano.shared(0., name="b")
```

Process to add/update packages

- ❖ We can easily add new full **configurations**
 - ❖ Although having a different configuration per each data scientist is not the desirable (difficulting sharing and reproducibility)
- ❖ The idea is to **agree** on the set of packages and versions
 - ❖ The user can always ‘extend’ a given configuration by adding new packages and versions installed locally (e.g. `pip install —user`)
- ❖ The natural forum to discuss and decide **configuration content and schedule** is the **LIM meeting** (librarians and integrators meeting)
 - ❖ Meets fortnightly
 - ❖ Representation from each LHC experiment, LCD, SWAN, etc.
 - ❖ IML representative should participate when required

Summary

- ❖ LCG configurations have been extended to fulfill, not only the requirement of ‘LHC productions’, but also final analysis
 - ❖ Many new packages have been added recently: Python, R, etc.
- ❖ LCG views is a simple and efficient way to get a **proper and reproducible runtime environment**
 - ❖ Very easy to build new custom specific ‘views’
- ❖ The idea is to extent the content of LCG releases with **more ML packages**
 - ❖ A number of ML package are already existing
 - ❖ New packages can be added (solving perhaps some technical problems)
 - ❖ Feedback most welcome
- ❖ The **LIM meeting** is the forum to discuss content and schedule