Introduction to CMS Open Data Analysis

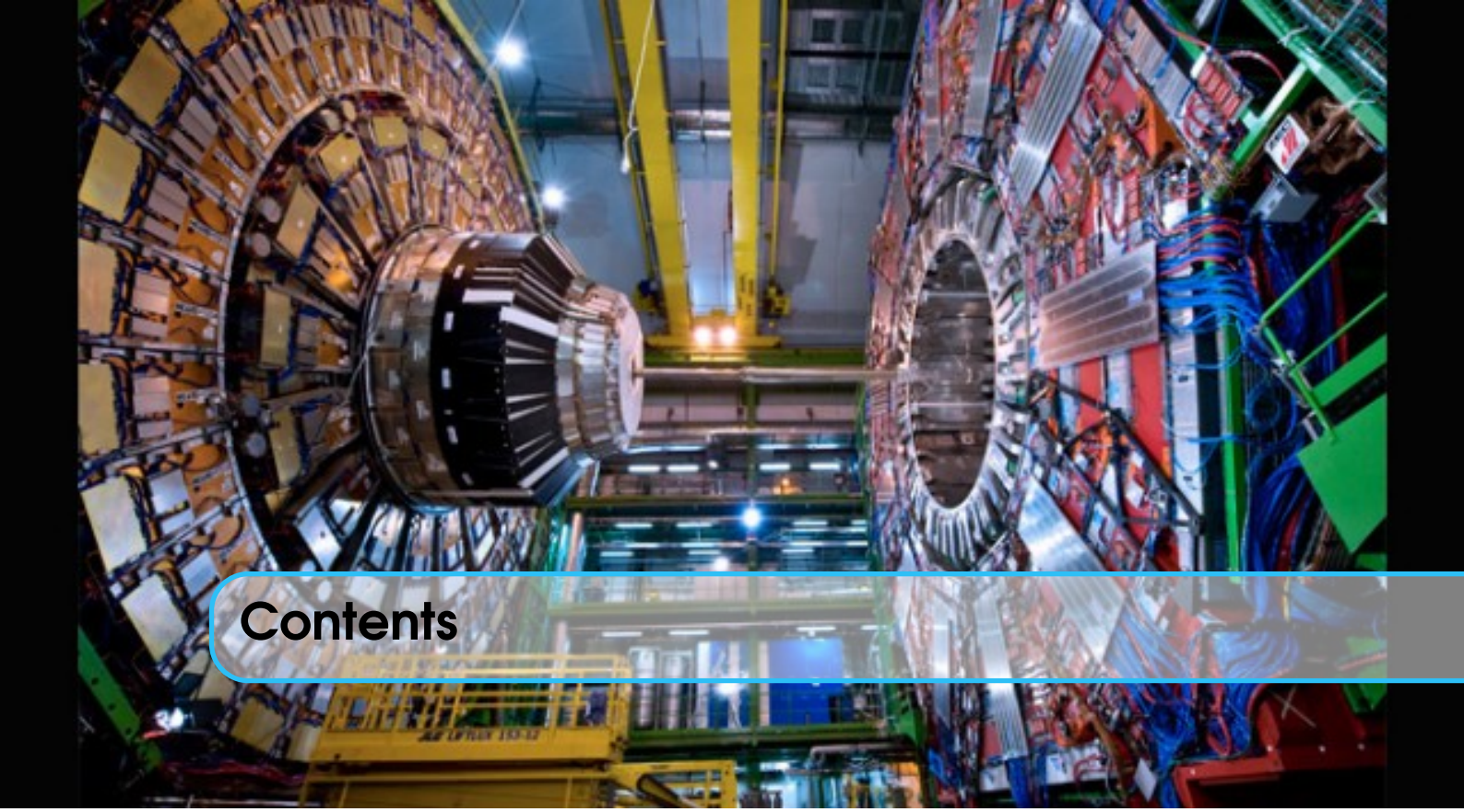
User's Guide for portal @ IFCA

G. Cañas / P. González

PUBLISHED BY G. CAÑAS AND P. GONZALEZ

https://github.com/Palmerina/CMS_OpenData_DidacticAnalyzer
https://github.com/palmerina/CmsOpenData_Analyzer

Version 1.0
October 2015



Contents

I

Quick Start-Up

1. Main objectives

1.1 Introduction

This book is a guide for students who desire to have their first contact in Particle Physics Analysis procedure. Basically, we introduce the reader into the Analysis of the CMS Open Data from 2010, with a focus on the Z Boson Decay into two muons.

The Analysis of the Z-Decay into two muons is performed trough a Software Package designed and written by Palmerina González Izquierdo. This Software Package is full written in Python. The description of this Software can be found in Section IV. **The Analysis**.

The description of the Cms Open Data Analysis from 2010 has been explained thanks to an Ontology in High Energy Physics with a focus on Cms Experiment designed by Guadalupe Cañas Herrera. The Ontology has been developed using *Protégé*. All information about the ontology may be found in Section V. **The Ontology**.

1.2 Level of Knowledge according to students type

This book is thought to be used by junior and senior students in the Physics Bachelor, so that they may encounter the most of the general content user-friendly. Nevertheless, the content of the book is clasified according the student level, and in Chapter 3 **Requirements**, the reader may find in each section what the expected academic level is.

In general, students' type may find the difficulty of the contect according to the following list:

- High-School science students: all this guide may be difficult to understand for this type of students, as they do not probably have the basic knowledge in particles physics or even in mathematical calculus. They do not have also previous learning in Computer Management, neither in programming. We suggest them to look carefully to all the chapter **Introduction to Computer Management**, and also to study

- First and Second year students from the Physics Bachelor: they will probably have enough programming skills to understand python codes and they may have taken basic lectures in Structure of Matter and even Particle Physics. However, part of the content refering to subsections 3.2, 3.3 and section 5 may be new for them. It is possible that they do not have a deeply knowledge in Particle Physics in order to fully understand the Analysis Procedure. In this case, we suggest them to have a look at . The part relationed to the High Energy Physics Ontology may be pretty difficult to follow without any experience in further computer sciences.

- Third and Fourth year students from the Physics Bachelor: the whole content of this guide, except chapter V and section 3.2 and 3.3 may be familiar to them. We expect them to use this guide and the software (booth the Analysis and the Ontology) to clarify concepts in Computer Management as well as in Particle Physics.

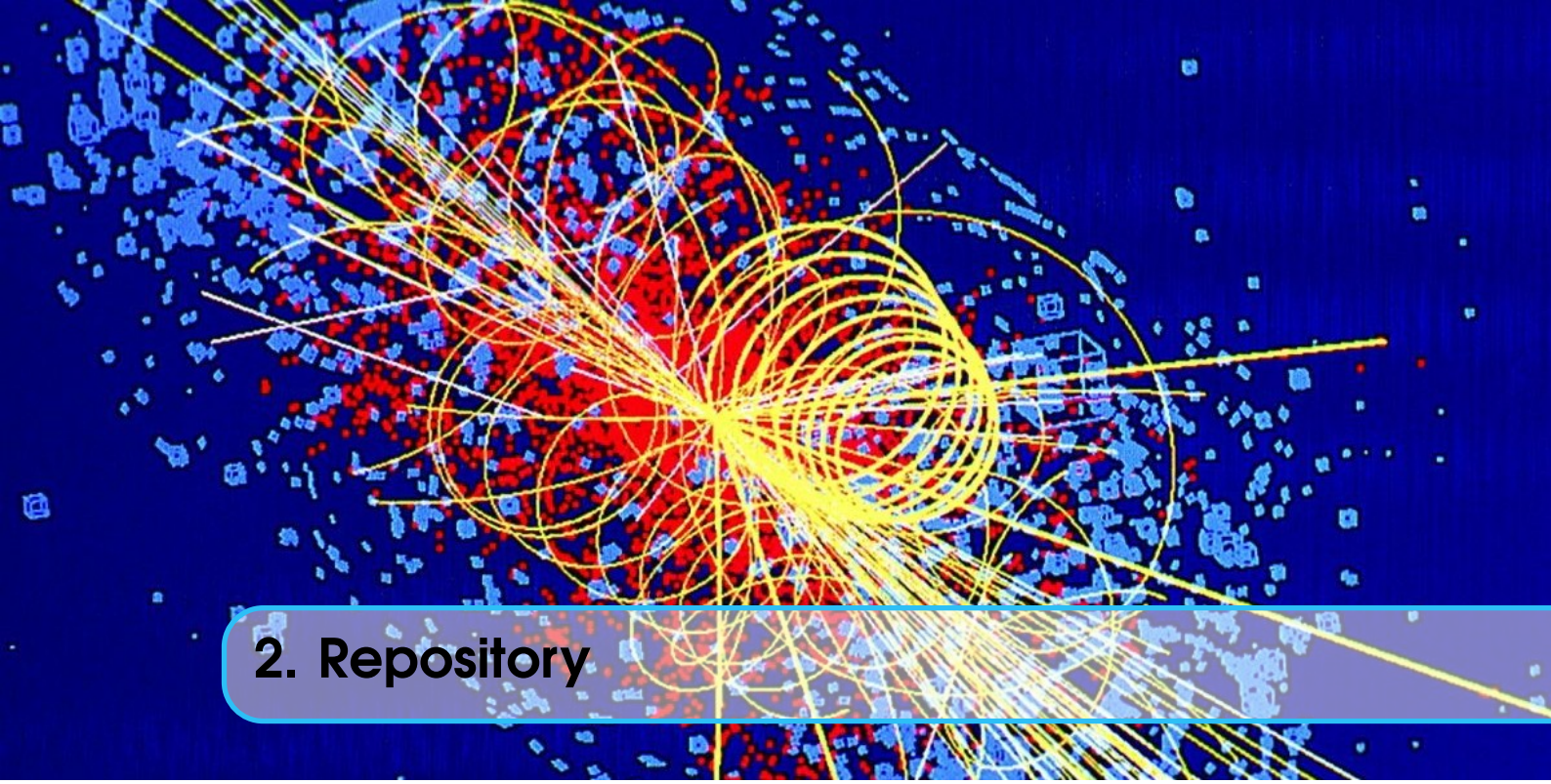
- Physics Master students: these students will probably be proficient at Computer Management and the part related to the Data Analysis may be found easy for them.However, the part explaining the Ontology could be completely new anyhow.

1.3 Where is the material?

All the software codes as well as the installations requirements are placed at different repositories:

1. Analysis Software: `https://github.com/Palmerina/CMS_OpenData_DidacticAnalyzer`
2. Ontology Software: `https://github.com/gcanasherrera/CmsOpenData_IFCA`

Introduction to Computer Management



2. Repository

2.1 Content of the Repository

All Software and codes are included in a web repository at `https://github.com/gcanasherrera/CmsOpenData_IFCA`.

The content of the GitHub repository is:

1. **README**: a text file that contains information about how to use the codes and different libraries or applications the user may need to run the programs.

2. **LICENSE**: a text file that includes information about the GNU license under which our programs are registered.

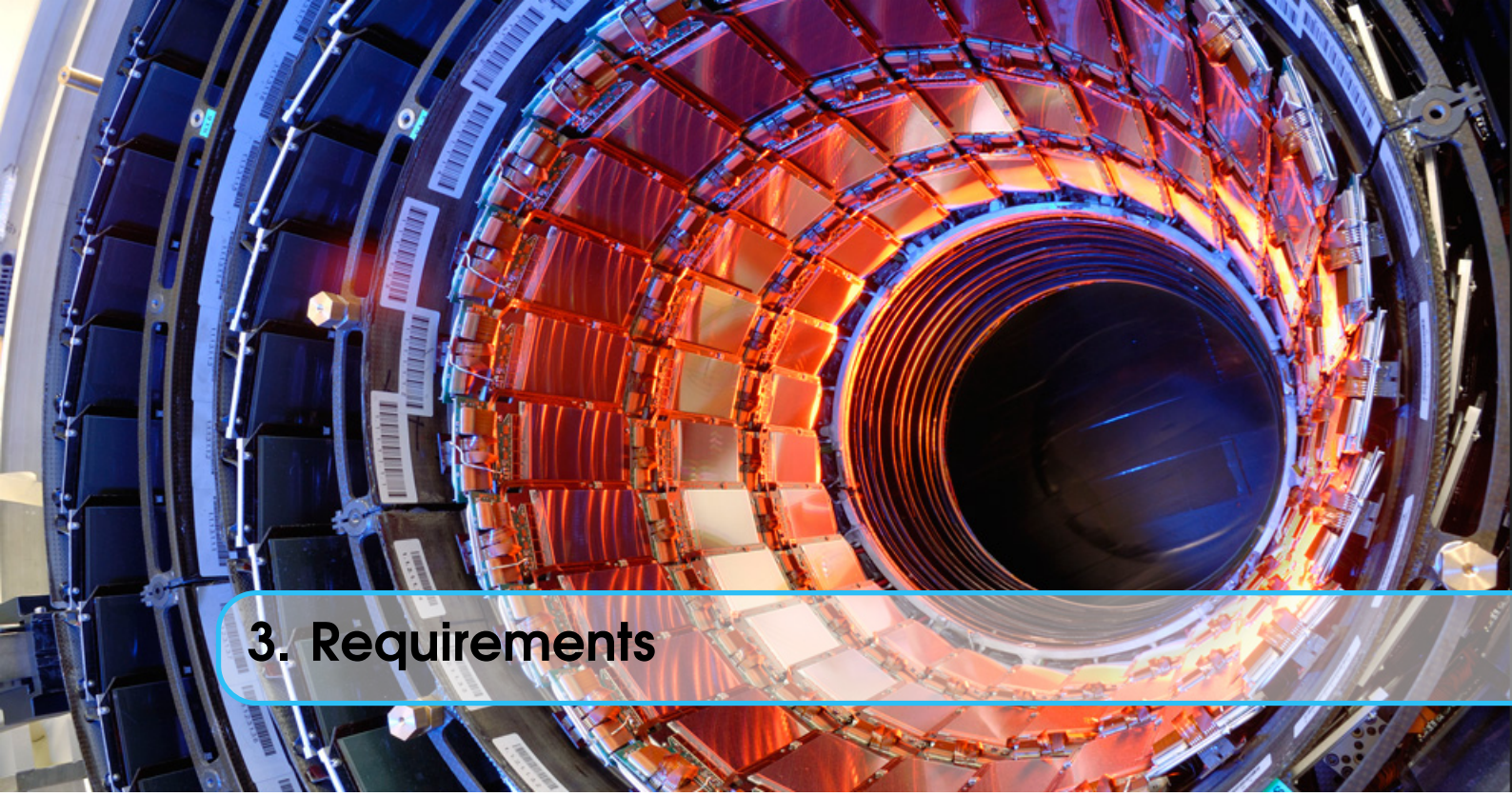
3. **CmsOpenData_Analyzer**: a directory that contains the Analysis Software Package with different Python codes.
   (a) *exercise1.py*: script that plots muons' magnitudes
   (b) *exercise2.py*:
   (c) *exercise3.py*:
   (d) *execute.py*: main script that runs the whole Analysis Software. In case the user run this script, it is not necessary to run above described scripts (exercise1.py, exercise2.py and exercise3.py).
   (e) *CutsConfig.py*:
   (f) *LeptonPair.py*:
   (g) *TwoMuonAnalyzer.py*:

4. **CmsOpenData_Ontology**: a directory that contains the Ontology codes.
   (a) *COD_Ontology.rdf*: rdf document that includes classified information about classes and individuals, together with data and object properties.

The full content of the item 3 and 4 are explained in chapters IV and V.

2.2 Future Updates

Future actualizations of the Analysis Software or the Ontology will be posted either in the Web Page (see Chapter II, section 4), or in the GitHub repository. This guide together with the README file will be also updated according to new improvements.



3. Requirements

CMS Data Analysis is very complex due to the large set of information collected in every collision. Therefore, it is necessary to use computers and software tools able to make theoretical and numerical operations easier. A good knowledge in Computing is highly recommended.

3.1 What you know

These are some explanations of basic concepts that most phyisicist have studied previously during some programming courses. In case you have studied in *University of Cantabria*, you can skip this chapter as these notions have been already tought in the *Programación* course. In case you are not familiar with the following topics, please take your time to be confident with them before keeping reading.

3.1.1 What an Operating System is

An Operating System (OS) is a computer software designed to operate and control the computer hardware and to provide a platform for running application software.

Nowadays, most people are *Microsoft Windows* users, which is a MS-DOS Operating System developed by Microsoft. However, when you are being introducted in science analysis, you realize that a great amount of scientists use **UNIX** based Operating Systems. The Unix-like family is a diverse group of operating systems with several major sub-categories including the famous *Linux* and its main distributions as *Ubuntu* or *Debian*. Apple's OS X Operating System is also Unix-like.

If the reader has never been in contact with an UNIX-based OS, starting to have a look at it is suggested. Most UNIX distributions are completely free and can be easily installed in every personal computer. It is convenient to remark that, for a non-experienced computer user, the main difference between Microsoft Windows and Ubuntu is the syntax of the commands written in the terminal (see next section).

From now on, it will be assumed that the reader knows how to manage himself in a UNIX-like OS.

3.1.2 What an Object-Oriented Language is

If you have never used an object-oriented programming (OOP) language before, you will need to learn a few basic concepts before you can begin writing or modifing any code.

Basically, the main dissimilarity with other non-OO, as for instance *C*, is the presence of a paradigm based on the concept of "objects", which are data structures that contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. Most popular OOP languages are class-based, in which object are instances of classes, which typically also determines their type. Some OOP languages include Python, C++, Java, Perl or Ruby. In the analysis of CMS Open Data, all programs have been written in Python.

3.1.3 What a Terminal is

A terminal is an interface in which you can type and execute text based commands.

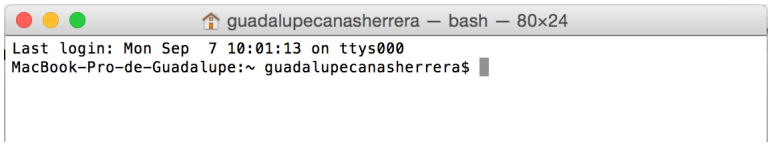


Figure 3.1: Screen Capture of a OS X Terminal just after being open.

It can also be known by the names Console or Shell. The terminal in LINUX-based OS is called bash. If you use a terminal you can complete some tasks using much faster than with graphical applications and menus. Another benefit is allowing access to many more commands and scripts.

There is a complete list of bash commands here `http://ss64.com/bash/`. The most useful ones you will probably use while you analyze the data through the already-prepared software are:

1. **cd**: this command allows you to change the directory you are placed at and ables you to move over different directories.
2. **ls**: this command shows you what is the content of a directory (different files or another directories).

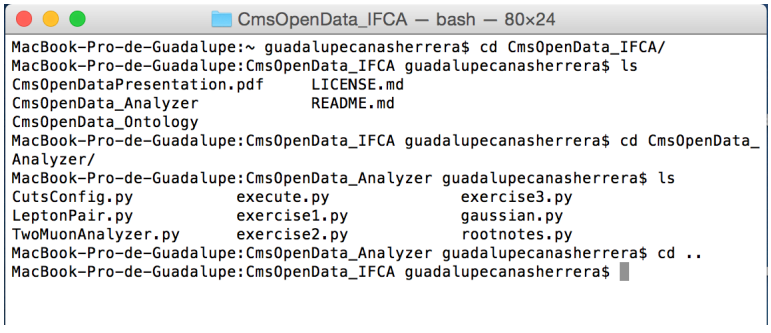


Figure 3.2: Screen Capture of a OS X Terminal using commands **cd** and **ls** along the directory **CmsOpenData_IFCA**.

In Figure 3.2. the reader may observe how the terminal user was placed in first instance at the HOME directory, called in this case **guadalupecanasherrera**. Through the instruction ***cd***

CmsOpenData_IFCA, the terminal user moves from the HOME directory to the directory under the name **CmsOpenData_IFCA**. Once the terminal user is at **CmsOpenData_IFCA**, using the command **ls** it is possible to show the content of the directory **CmsOpenData_IFCA**. There are two directories (**CmsOpenData_Analyzer** and **CmsOpenData_Ontology**) and two text files. Again, using **cd**, the terminal user moves into the directory **CmsOpenData_Analyzer** and shows again the content using **ls**. Finally, the terminal user moves one step up into the directory **CmsOpenData_IFCA** using the command and its attribute**cd ..**.

Using a bash terminal anybody can open a **text editor** in order to modify text files without going out of the terminal itself. We suggest using a text editor called **vim**, which is available in most UNIX-based OS. The reader is kindly referred to section 3.2.1 to find more information about this program.

3.2 What you should know

In this section we have included some topics that may be familiar for some, but also may be completely indistinct for others. Those vague ideas that the reader could have are here clarified in order to proceed with more important topics.

3.2.1 What Git is

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. A version control system is in charge of the management of changes to documents, computer programs, large web sites, and other collections of information. To download and install git, please be referred to `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`.

You will need Git in order to have contact with the software, as it is placed in a virtual git repository known as **Github** (web-site `https://github.com`). Basic commands for git use can be found at this cheat sheet `https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf`.

If you still need further information about git, the official git book is at `https://git-scm.com/book/en/v2`. We strongly suggest taking your time to learn hot to use git, as it will be pretty useful in your programming life in the future.

3.2.2 What Vim is

Vim (a contraction of Vi IMproved) is a text editor for Unix. Vim's interface is not based on menus or icons but on commands given in a text user interface; its GUI mode, gVim, adds menus and toolbars for commonly used commands but the full functionality is still expressed through its command line mode.

Vim has a built-in tutorial for beginners (accessible through the "vimtutor" command). There is also the Vim Users' Manual that details Vim's features. This manual can be read from within Vim, or found online (`http://vimdoc.sourceforge.net/htmldoc/usr_toc.html`).

Vim also has a built-in help facility (using the :help command) that allows users to query and navigate throughcommands and features.

For the purpose of this exercise, you only need to know these few things:

1. **Opening the file:** for example, if you want to edit the file execute.py, you just have to type on the terminal: *vim execute.py*

2. **Moving around:** you cannot move the cursorusing the mouse, but the arrow keys. You can also type "gg" or "G" to go to the start or the end of the file, respectively
3. **Editing text:** to delete and add characters as you would normaly do it in a GUI editor, you just have to type "i" and start editing. Press *Esc* to stop editing and use again the Vim commands.
4. **Searching for a word:** for example, if you are looking for th word "muon", you just have to type: */muon*. The word will be automatically highlighted. To move to the next time it appears, type "n".
5. **Exit Vim:** type: *:q*. To exit saving the changes applied, type: *:x*.

3.2.3 What a library is

In programming, a library is a collection of precompiled routines that a program can use. The routines, sometimes called modules, are stored in object format. Libraries are particularly useful for storing frequently used routines because you do not need to explicitly link them to every program that uses them. The linker automatically looks in libraries for routines that it does not find elsewhere.

During CMS Data Analysis, several libraries are imported depending on programming language is being used. In this case, we will focus on Python libraries and packages. Further information about Python Libraries will be provided during *The Analysis* part.

3.2.4 What a Virtual Machine is

In most cases, your personal computer is not prepared to run some codes or even softwares as you lack of different packages, libraries or even the appropiate OS. In those situations, the most suitable procedure is not to change your computer from scratch but using an emulation of a particular computer system. This software that enables one computer system (called the host) to behave like another computer system (known as guest) is what we name Virtual Machine (VM).

In order to run a program package able to analyze CMS Data we will need to use a VM that reproduces the whole environement of a UNIX OS with ROOT and other CMS libraries and packages.

3.3 What you do not (probably) even know

3.3.1 What ROOT is

ROOT is mainly used by the scientists at CERN in order to analyze the huge amount of data collected by the detectors. It is a modular scientific software toolkit and provides all the functionalities needed to deal with big data processing, statistical analysis, visualisation and storage. It is mainly written in C++ but well integrated with other languages such as Python and R. To sum up, it is a super *MATLAB* designed for particle data analysis.

For all the information about ROOT, please refer to the ROOT website: `https://root.cern.ch/`

3.3.2 What CMS Framework Light is

CMS Framework Light (CMS FWLite) is just a ROOT session with CMS data format libraries loaded. Moreover, CMS provides a couple of classes that greatly simplify the access to the collections of CMS data objects. These classes are called Event and Handle. It is required to use these classes when working with Patuples.

For further information about CMS FWLite, please visit `https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookFWLite`

3.3.3 What an Ontology is

An Ontology is a collection of concepts and its relations, meaning linked data, used precisely in a domain of knowledge through a common conceptual vocabulary. Ontologies are machine readable, so that they allows to ease data reuse and annotation. The use of an Ontology promote knowledge sharing and they are being increased in research combine with workflows to semantically describe processes. Colloquially, it may be said that an Ontology is a simple organized scheme of a topic that can be understood both by people and computers.

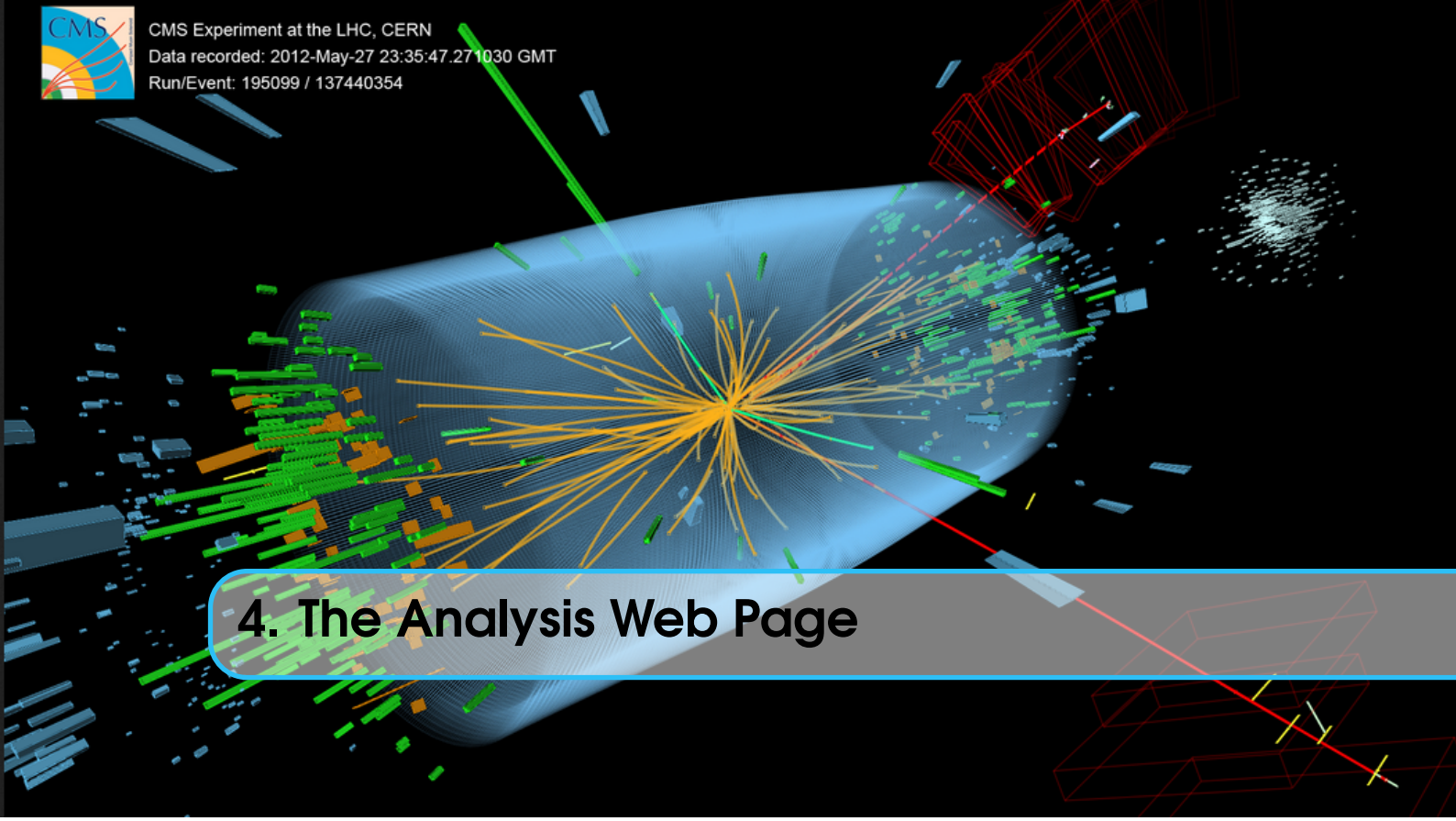
3.4 Requirements Summary

To sum up, to analyze CMS 2010 Open Data you will need:

1. A virtual Marchine (VM) with ROOT and CMS Framework Lite
2. Basic knowledge of bash commands
3. Basic knowledge of VIM commands
4. Basic knowledge in Object-Oriented Programming

On the other hand, in order to work with the High Energy Physics Ontology:

1. A user-friendly Ontology interface and editor, as *Protégé* (more information in section V).
2. Basic knowledge of SPARQL (see chapter).
3. Basic knowledge in Ontology vocabulary.



4. The Analysis Web Page

The analysis of the data is developed thanks to a web page: `https://cmsopendata.ifca.es`. In this web-page, you will have online access to a Virtual Machine (VM) that contains everything required to run your Analysis of the CMS 2010 Open Data.

Figure 4.1: Screen Capture of the Analysis Web Page `https://cmsopendata.ifca.es`.

At the moment of entering into the Web Page, you will need to register before starting your analysis. For that, you must click into the bottom *star your analysis* at the main page, and then into the bottom *Request an account*. Then, you have to fulfill your username, e-mail and password.

Once you have you account, you are able to *sign in* with your new username. You will see a

screen like the one in Figure 4.2. At this moment, you can start to run exercises and programs.

Figure 4.2: Screen Capture of the Virtual Machine Terminal shown at `https://cmsopendata.ifca.es` once you have signed in.

5. Connection through ssh

It is possible to use the already-prepared Virtual Machine available at `https://cmsopendata.ifca.es` without going explicitely into the web-page itself. There is another way based on a **ssh connection**.

The ssh connection, or Secure Shell connection, is a cryptographic (encrypted) network protocol to allow remote login and other network services to operate securely over an insecure network. Basically, we are going to force our bash terminal in a LINUX-based OS to work as the Virtual Machine you may observe at the web-page.

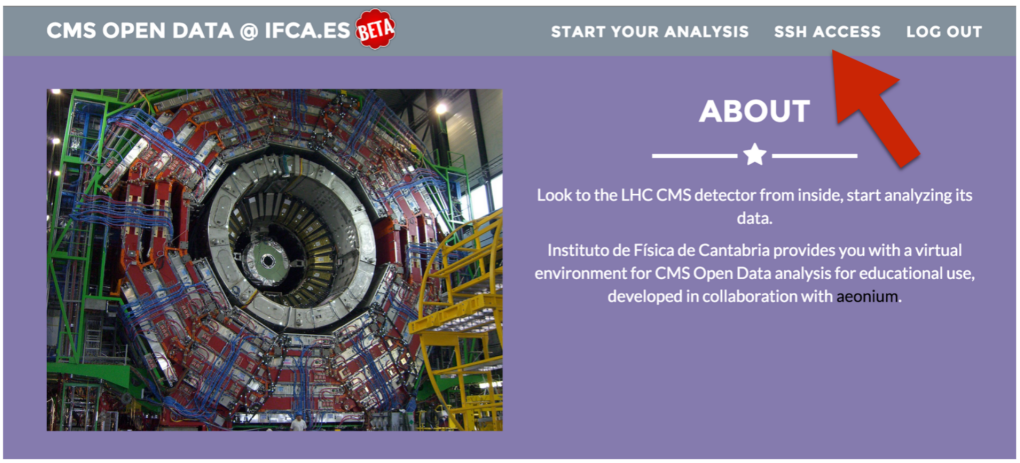


Figure 5.1: Screen Capture of the main view at `https://cmsopendata.ifca.es` once you have already signed in.

5.1 Going into the ssh Web Page Menu

To connect via ssh, you will need to create a key that ables you to connect your personal computer, and therefore your terminal in your own OS, with the already prepared Virtual Machine.

- First, you need to log in with your username and password, and then you must click on the bottom *ssh access* as it is shown in Figure 5.1. You will go to a new page that looks like the one observed in Figure 5.2.
- This menu shows you what your **ssh user** is in relation to your username you created before when you got registered. In this case, the ssh user that is **cms0056**. Note that every reader of this guide will have a different ssh user.
- You will also observe that there is a big square blank space under the name *Public Keys*. You will need to enter your public key in this space.

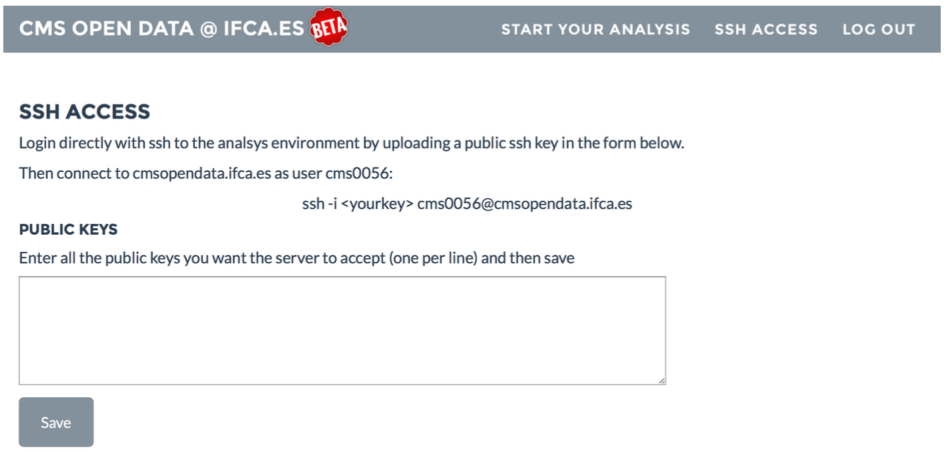


Figure 5.2: Screen Capture of the ssh view at `https://cmsopendata.ifca.es` once you have already signed in.

5.2 Creating the public key

In order to create a public key in your computer and to establish a relation between that public key and your ssh user you will need to procedure and to type some commands in the terminal of your computer. First at all, you need to open a terminal in your own computer and follow the next instructions.

5.2.1 Check for ssh keys

We need to ckeck whether there are pre-existing ssh public keys in your computer. For that, you need to type on the terminal the command **ls -al ∼/.ssh**.

If you receive an error that *ssh* doesn't exist, don't worry! This means you do not have any ssh key and we will create it in the following step (see Figure 5.3).

5.2.2 Generate a new SSH Key

With the terminal still open you must type (or copy and paste) the next command: **ssh-keygen -t rsa -b 4096 -C "your_user@cmsopendata.ifca.es"**. Your_user is the ssh user you observe in Figure 5.2.

At the moment you type the command **ssh-keygen**, some questions related to the file where the ssh key will be save pop up. We suggest keeping the default settings to avoid problems in the future, so when you are prompted to "Enter a file in which to save the key", just press *Enter* to continue.

Next, you will be asked to write a passphare to save your SSH key. You can decide yourself if you want to include a password or not. At the end you will see the key fingerprint you have created (see Figure 5.3).

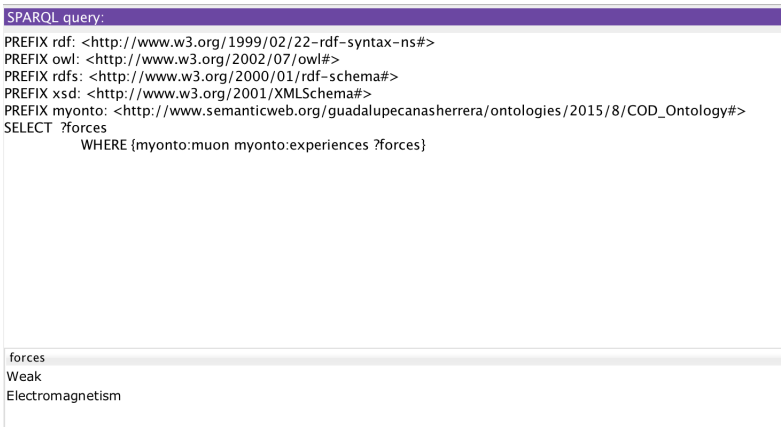


Figure 5.3: Screen Capture of a LINUX terminal checking for ssh keys and generating a new SSH Key. Observe that no public keys were found when the ls command was typed.

5.2.3 Add your key to the SSH-agent

With the key already created, you are able to configure the SSH-agent program to use the SSH Key. You will need to type the commands **eval "$(ssh-agent -s)"** and **ssh-add ~/.ssh/id_rsa**

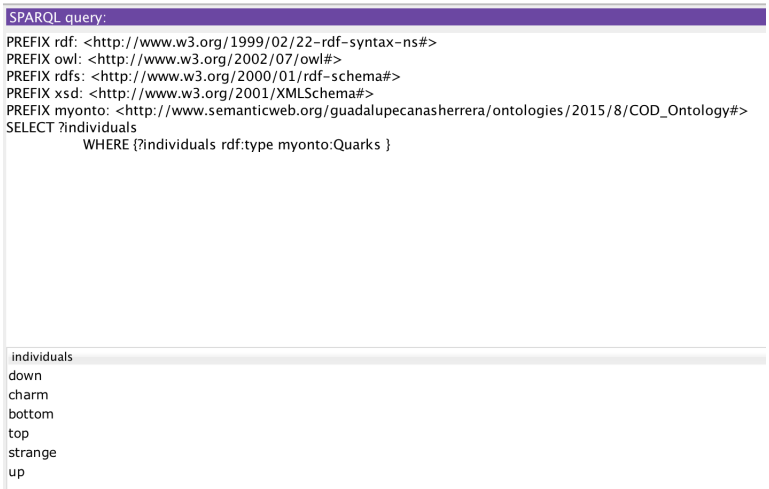


Figure 5.4: Screen Capture of a LINUX terminal adding the public key to the SSH_agent.

5.2.4 Add your SSH key to the Analysis web page

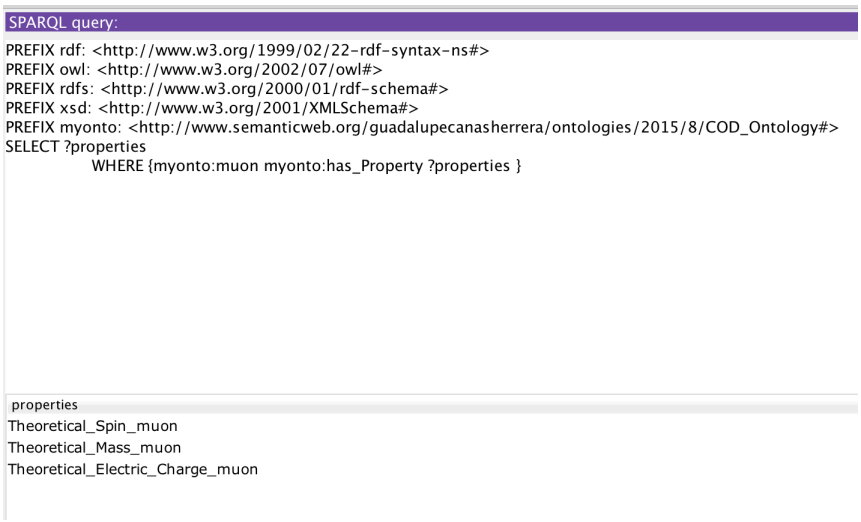


Figure 5.5: Screen Capture of a LINUX terminal showing the public key through vim.

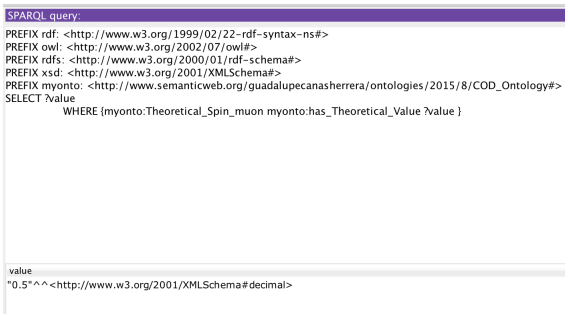


Figure 5.6: Screen Capture of the analysis web page with the SSH key already pasted into the blank space.

First, you will need to move to the directory where the public key is contained using **cd /.ssh** and open with vim the file *id_rsa.pub*. For that, you must type **vim id_rsa.pub**. Simply copy the text shown in vim using your computer mouse.

After copying the key, go again to the analysis web page at the SSH Access view as it is shown in Figure 5.2, and paste the key in the blank space under the name public Key.

5.3 Acces from your terminal using the public key

At the end, you will be able to open a new terminal and type the command line: **ssh -i id_rsa.pub cms0056@cmsopendata.ifca.es**. In case you decided to include a password when creating your SSH public key, it will be asked after pressing enter.

Voilà!, you can connect know to the VM without going into your internet browser, just using the terminal. In that case, you just need to open a new terminal and tab on it: **ssh -X cms0056@cmsopendata.ifca.es**. The command **-X** enables the possibility of popping out plots and graphs in new windows directly from the VM. This will be required in order to look at the different plots of the particles magnitudes a the moment of analyzing the data.

III Physics Background

6. General overview

6.1 The Z boson

6.2 Z production

- **Primary vertex:** point where the particles collide.
- **Secondary vertex:** point where the product particle is produced. It corresponds to the first hit of a particle's track.
- **Hits:** when a particle crosses the detector, it produces electric signals on its path, which correspond to the hits.
- **Tracks:** reconstructed particles' path from the hits.
- **Impact parameter:** it is the perpendicular distance from the particle's track to the primary vertex.
- **Transverse distance:** the distanze from the secondary vertex to the primary one projected into the z direction (respect to the beam or detector's axis).
- **Transverse momentum:** particle's momentum projected into the z direction (respect to the beam axis).
- **Pseudorapidity (η):** it is a commonly used spatial coordinate describing the angle of a particle relative to the beam axis. It is defined as:

$$\eta = -ln\left[tan\left(\frac{\theta}{2}\right)\right] \tag{6.1}$$

Where θ corresponds to the angle between the particle three-momentum **p** and the positive direction of the beam axis. This is the same as saying that η is similar to the angle between **p** and the transverse direction of the beam, as we can see in Fig.6.2.

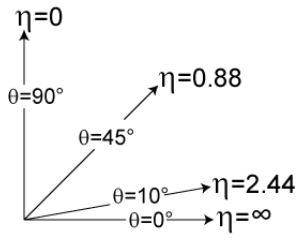


Figure 6.1: Pseudorapidity and θ angle respect to the beam axis.

We will look for the muons with low η (detected in the barrel, which is the central part of the detector), which correspond to the most energetic ones and, therefore, the best candidates to come from a Z decay.

- **Jets:**

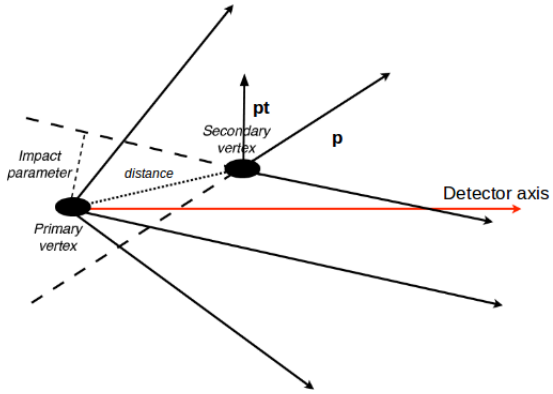


Figure 6.2: Variables involved in a CMS event.

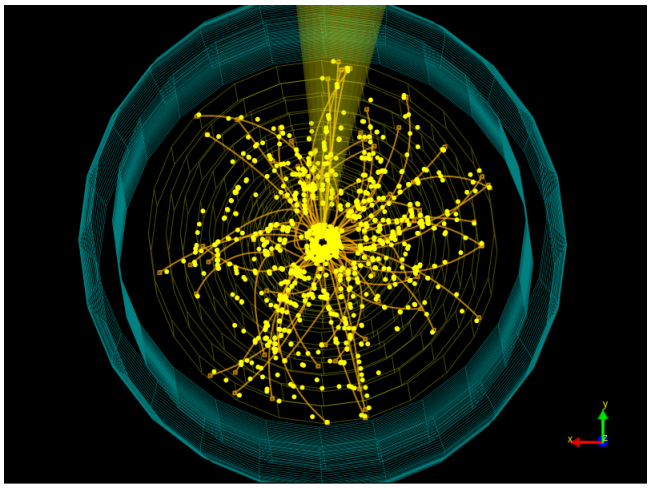


Figure 6.3: Simulation of part of the CMS detector, showing the reconstructed hits, tracks and one jet of particles.

6.3 Particle detectors

6.3.1 CMS: Compact Muon Solenoid

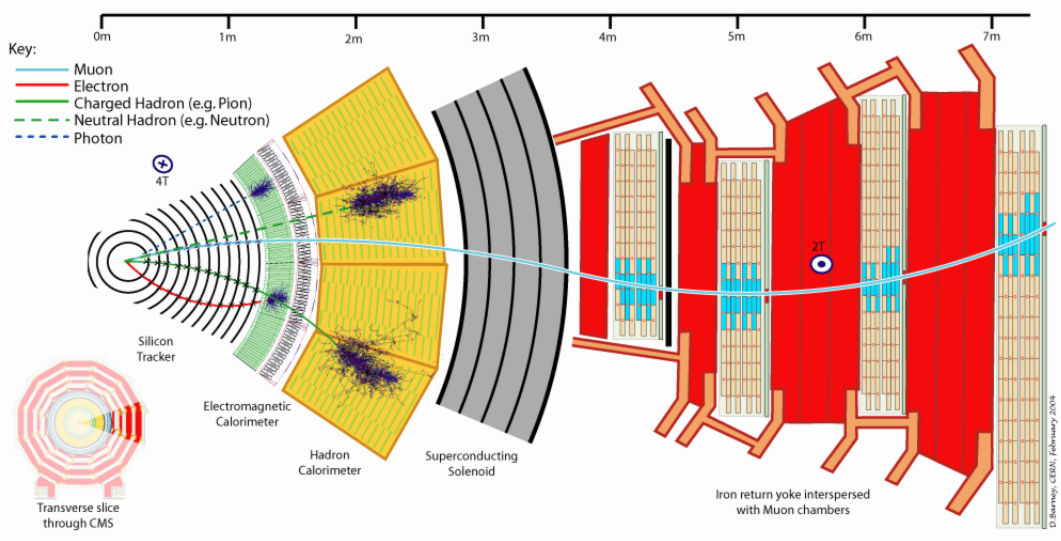


Figure 6.4: Scheme of the CMS Detector and the tracks left by the different particles.

IV The Analysis

7. The Analysis Software

The analysis software developed for didactic purposes is contained in the *CmsOpenData_Analyzer* package. This package has six files which are: TwoMuonAnalyzer.py, LeptonPair.py, CutsConfig.py, execute.py, exercise1.py, exercise2.py and exercise 3.py.

You can find all the code on the repository: `https://github.com/Palmerina/CMS_OpenData_DidacticAnalyzer`.

7.1 Classes

7.1.1 TwoMuonAnalyzer

This class is placed in TwoMuonAnalyzer.py and is the one who analyzes the data. First of all, it imports the data (muons and primary vertex) with the classes Handle and Events from DataFormats.FWLite.

The class' function that analyzes the data is process(), which loops over all the events and then loops over all the muons and adds their variables to arrays (class' attributes).

Then, it applies the selection cuts (function selecMuons()) to choose the good muons (those coming from the Z decay). Again, it adds their variables to arrays and paires them up. To do this, there ir another loop over the muons and paires up those which have opposite charge. Then, a LeptonPair object is created with these two muons and the primary vertex and it gets its mass, adding it to another array.

The functions plotter and plotter1 plots the histograms created in process.

7.1.2 LeptonPair

This class must be called with three parameters: l1, l2 and vertex, which must be two pat::Muon objects and a reco::Vertex object, respectively.

The function of the class is to pair up both muons and to obtain the pair's invariant mass, transverse momentum and distance to the primary vertex.

7.1.3 CutsConfig

This class has just a constructor whose attributes are the values which will be used in the muons' selection process.

7.2 Exercises

There are three exercises, each of them plots different variables and it is in a separate script. You can also plot everything at the same time executing *execute.py*, which contains the three exercises (this script is thought to be executed in an ipython notebook).

Every exercise contains:

- **Data:** the data consists in PAT tuples (PAT stands for Physics Analysis Toolkit) in .root files. To know more about PAT tuples and how they store the information, take a look to `https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookPATGlossary`
- **maxEv:** maximum number of analyzed events. To obtain the Z boson peak at least 1000000 events are needed. This execution lasts approximately 15 minutes.
- **CutsConfig's object** called *cutsConfig*.
- **TwoMuonAnalyzer's object** called *analyzer*.
- **TwomuonAnalyzer's function:** a different one for each exercise and the three of them in*execute.py*

7.2.1 exercise1.py

This exercise uses the TwoMuonAnalyzer's function *plotter1()*, which plots several variables of all the muons without selecting the good ones.

These variables are:

- **pt_1 and pt_2:** transverse momentum of each muon of every pair (Fig. 7.2.1).
- **Dz to PV:** distance of the muon vertex to the primary vertex in the z axis respect to the detector's frame of reference (Fig. 7.2.1).
- **Impact parameter:** perpendicular distance between the muon's track and the primary vertex (Fig. 7.2.1).
- **chi2:** track's χ^2 (Fig. 7.2.1). It gives us an idea of how well fitted are the reconstructed tracks to the particle's hits on the detector: the lower the χ^2 is, the better the fit is.
- **Number of valid hits:** muon's number of valid hits in both the tracker and the muon chambers (Fig. 7.2.1).

7.2.2 exercise2.py

This exercise uses the TwoMuonAnalyzer's function *plotter()*, which plots several variables of the selected muons within all the muons without selecting the good ones.

This exercise contains, besides the already mentioned components, several CutsConfig parameters which must be changed until the right values which make you get the Z boson peak are found (Fig. 7.2.2).

These variables are:

- **pt_1 and pt_2:** transverse momentum of each good muon of every pair (Fig. 7.2.2).
- **Invariat mass:** distance of the muon vertex to the primary vertex in the z axis respect to the detector's frame of reference (Fig. 7.2.1).
- **Total pt:** perpendicular distance between the muon's track and the primary vertex (Fig. 7.2.1).
- **eta:** muon's pseudorapidity (see section: 7.4)(Fig. 7.2.2).

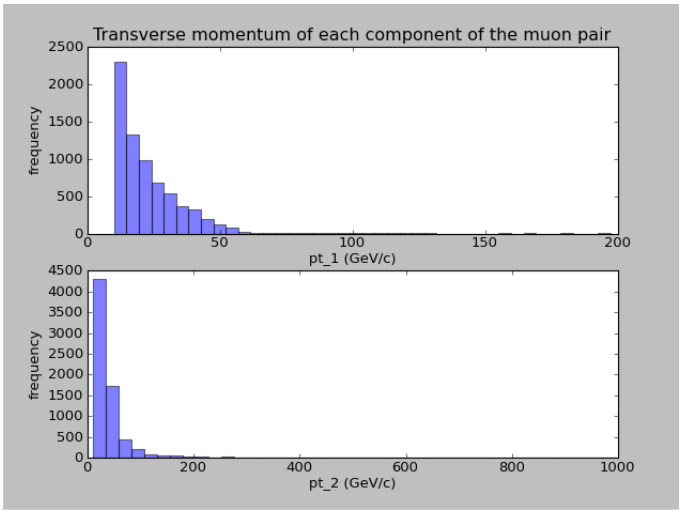


Figure 7.1: Histograms of the transverse momentum of each muon of every pair.

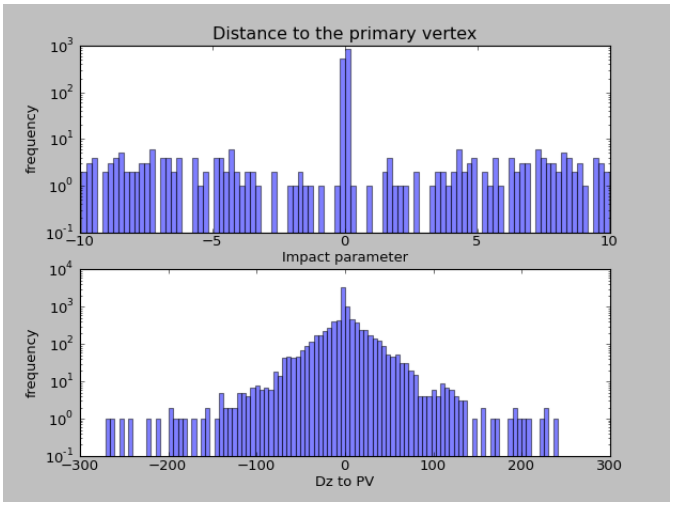


Figure 7.2: Histograms of the impact parameter and the distance of the muon vertex to the primary vertex in the z axis.

7.2.3 exercise3.py

I fits the Z boson peak to a gaussian and a Breit-Wigner function. The theoretical curve to describe the peak is the Breit-Wigner function, so this one should fit better.

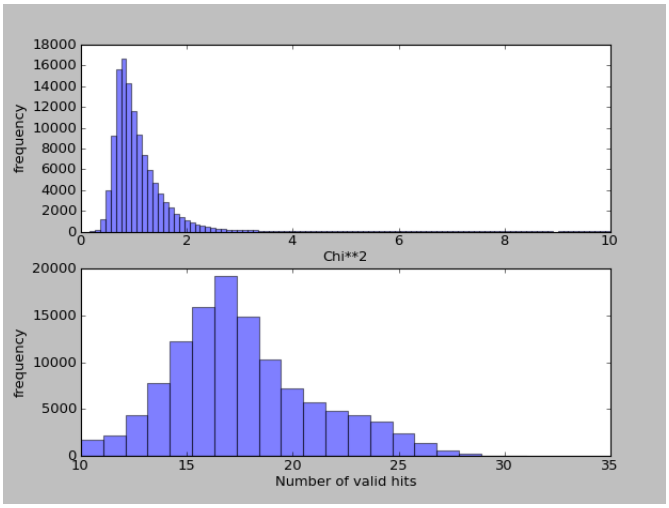


Figure 7.3: Histograms of the track's χ^2 and the number of valid hits in both the tracker and the muon chambers.

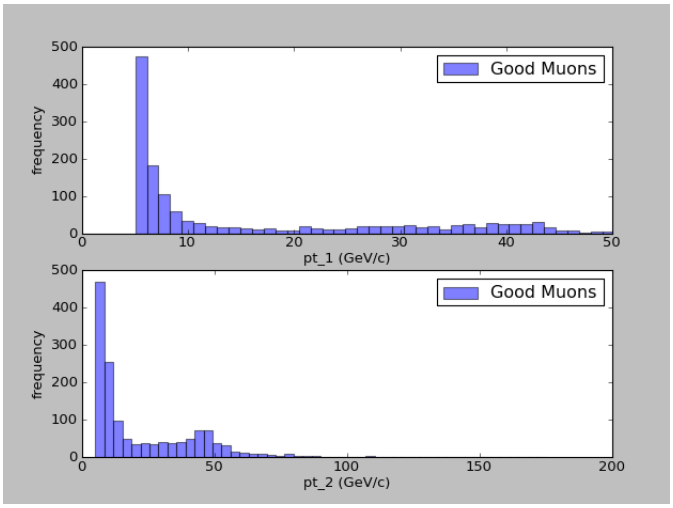


Figure 7.4: Histograms of the transverse momentum of each good muon of every pair.

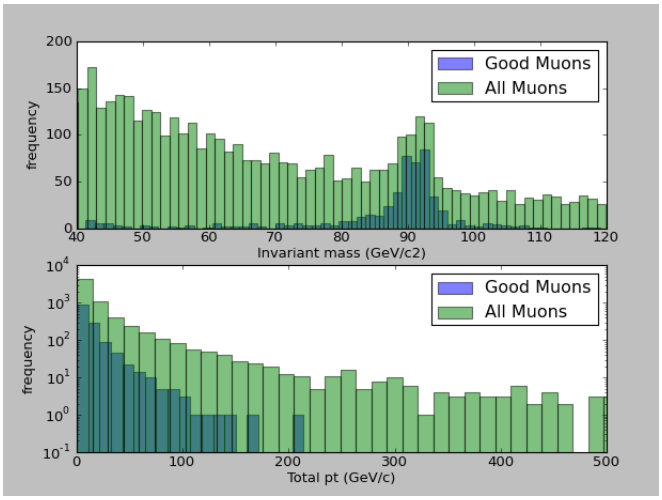


Figure 7.5: Histograms of the invariant mass and the total transverse momentum of each pair of muons.

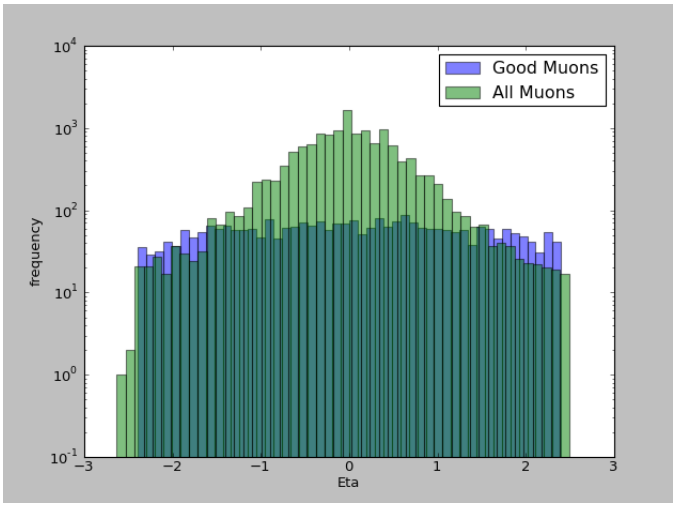
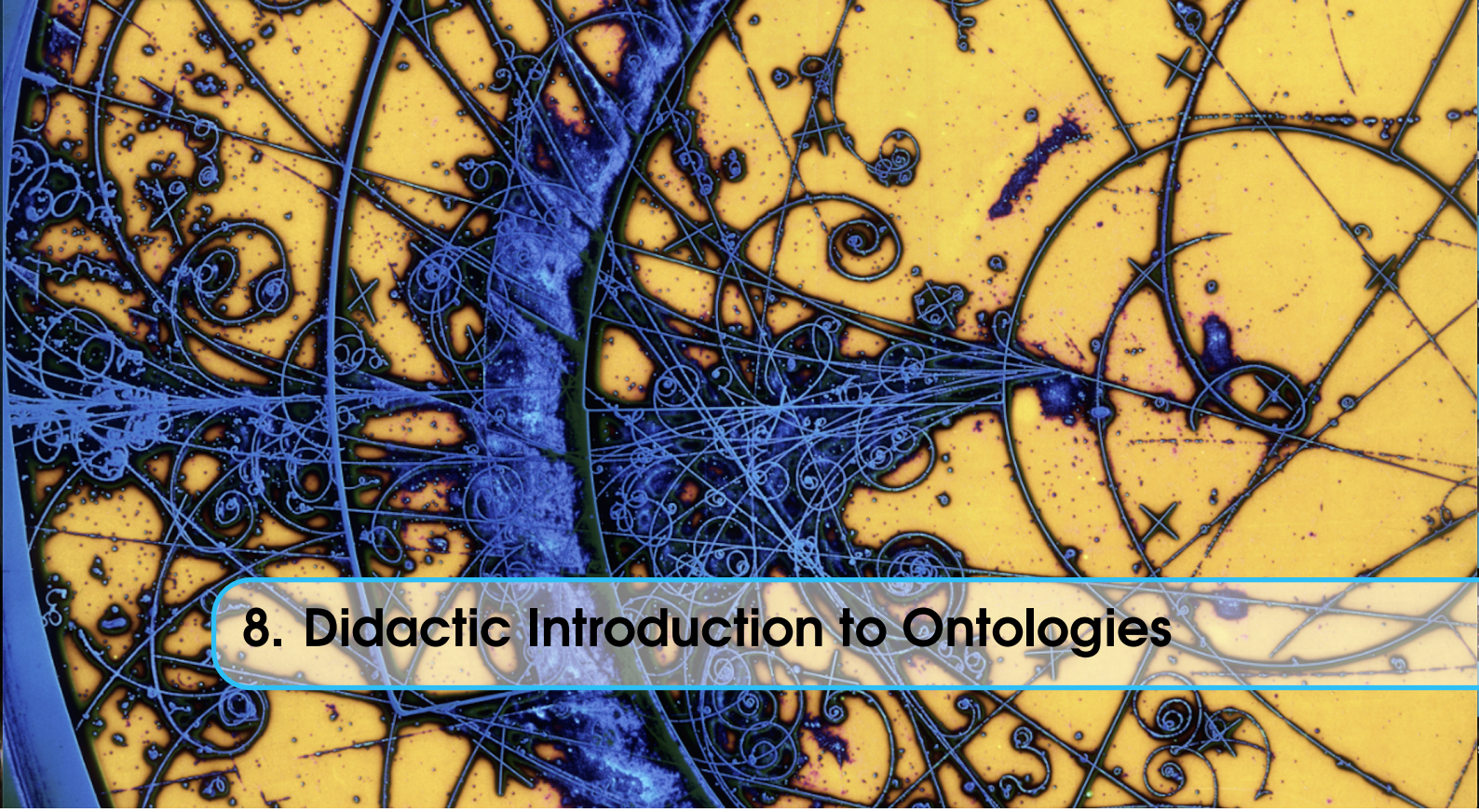


Figure 7.6: Histograms of the muon's eta angle.

V

The Ontology



8. Didactic Introduction to Ontologies

Accoding to Tom Gruber, an ontology is *a formal specification of a shared conceptualization*. This definition is widely spread, although its meaning may be complex and abstruse for somebody who has just started to learn about this topic.

An Ontology could be understood as *a set of concepts in a domain of knowledge linked by their relations through the use of a common vocabulary*. Therefore, an Ontology is basically a model that describes a concrete knowledge field and introduces the possibility of organizing and planning science research. This scheme can be understood both by people and machines, easing the way of communicating among them. At the same time, the communication allows sharing knowledge.

8.1 Learning the concepts

The first step before going any further is understanding some concepts in Computer Sciences related with Ontologies. Basically, computer resources, which are any physical or virtual components of limited availability within a computer or information management system, are identified through Uniform Resources Identifiers, or URIs, according to the Semantic Web and the World Wide Web Consortium. To organize these URIs, it is used the Resource Description Framework (RDF), that allows also to describe relations between URIs in the form of triples. The triplets are in the form of,

$$\textit{Subject} + \textit{verb} + \textit{complement}$$

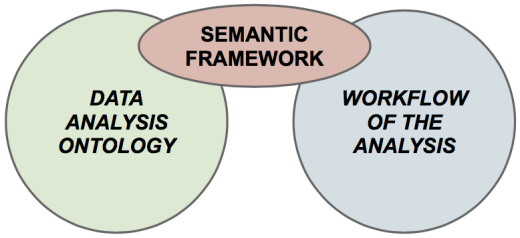
where the subject is a URI and the complement may be another URI or information related to the subject as data. In this last case, that data associated to the *subject* is known as *metadata*.

The organization of these triplets may be applied in a common way in order to share knowledge. In this case, the *verb* in the triplet structure is obtained from a common language. The most well-known are RDF-s and OWL. Furthermore, OWL introduces the possibility of organizing the triplets through logical closures so that we can include conditional situations. Therefore, RDF

together with RDF-s and OWL provide the perfect mechanism for designing an ontology in a field of knowledge.

8.2 Why to use an Ontology?

Most of the times, researchers works with already-defined steps in procedure. Data Analysis processes generally consist in a sequence of time ordered steps over data suitable to be modelled using scientific workflows. Consequently, a scientific workflow is a description of a process for accomplishing a scientific objective. If we develop an ontology able to express the logic between concepts and step in the process, we will obtain an ideal way of implementing a workflow thanks to a general semantic framework; that is, a collection of common vocabulary and expressions in a knowledge field.



The Ontology Implementation may also promote data analysis preservation, including the data itself as well as the procedure to obtain the final result. It can be also useful for new students being introduced in a field where a workflow is already on use.

8.3 Getting prepared with Protégé

The best (and easiest) way of designing an ontology is through the proper graphical tool. In this case, it is suggested going through ontologies thanks to *Protégé*. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. It was developed by Stanford University and has become one of the most famous ontologies' application.

One of the best features of last *Protégé* version, meaning 5.17.0, is the fact that it only requires a a Java Runtime Environment to run platform independent software. This implies that you can work with this version independent of the OS you have installed in your lapton, you just need to have a Java Runtime Environment. If you do not have this Java package, you may obtained at `https://www.java.com/es/download/`. Remind that you will need to have last Java version in order to run Protégé version 5.17.0. Also may the reader take into account that this Protégé version is still a Beta. Nevertheless, it is stable enough to work with at a didactic level. To download this version independent of your OS you have to visit `http://protege.stanford.edu/products.php#desktop-protege` and click at Download platform Independent version.

After downloading, to run the application you have to:

- In LINUX: open a terminal, go into the Protégé folder you have downloaded using the command **cd**, and run the bash script run.sh with the command **sh run.sh**

- In OS X: open the folder you have just download, and double-click on the run.command file.

If an error appeared stating that it was not able to open the program, I would have to check your JAVA version and environment.

8.3.1 Creating your first ontology: a tutorial

Before going into the ontology about the CMS Open Data in High Energy Physics, it is convenient to get familiarized with Protégé and other ontologies first. There are several tutorials on the internet about already-made ontologies: However, the author suggests following this example about *pizzas* at `http://protegewiki.stanford.edu/wiki/Protege4Pizzas10Minutes` and create yourself your first ontology. In this way, you will be forced to interact with Protegé and think about how an ontology is designed.

In case you prefer following a video-tutorial, it is advised to following this one at youtube `https://www.youtube.com/watch?v=R9ERlUgvgwM$&$list=PLea0WJq13cnAfCC0azrCyquCN_tPelJN1` by Dr Noureddin Sadawi. Furthermore, if you are really interested in managing properly Protégé, Stanford University sometimes offers free courses (check-out its web-page `http://protege.stanford.edu/short-courses.php`).

Creating an Ontology is pretty similar to learn programming skills: you will achieve them if you practise, so don't be afraid and get working!.

8.4 The parts of an ontology

As it is been already mentioned, an ontology is a formal explicit description of concepts in a domain of discourse in the form of **classes**, **properties** of each class describing various features and attributes of the concept, and restrictions on properties known as **facets**. An ontology together with a set of **individual** instances of classes constitutes a **knowledge base**.

The concept of ontology is quite similar to Object-Oriented programming. Therefore, individuals can be thought to be as instances that inherit their main properties from a fatherly class, which also include another properties that identify only that individual. This analogy with OO programming is explained in Table 8.1.

	Object-Oriented Programming	**Ontology**
General Part	Class	Class
Individual Part	Object	Individual
Properties	Attributes	Data Properties / Object Properties

Table 8.1: Visual explanation of the similarities between Object-Oriented Programming and

Individuals are related with another individuals or metadata through properties that are classified as,

- **object properties**: for instance,

  experimental_muon is_detected_in muon_chambers

  where experimental_muon is a individual from the class Events and muon_chambers is a individual from the class Analysis.

- **data properties**: for instance,

  CMSOpenData_Analyzer has_Version "1.0.0"

  where CMSOpenData_Analyzer is a individual from the class Software and "1.0.0" is simply a string value attached to the individual.

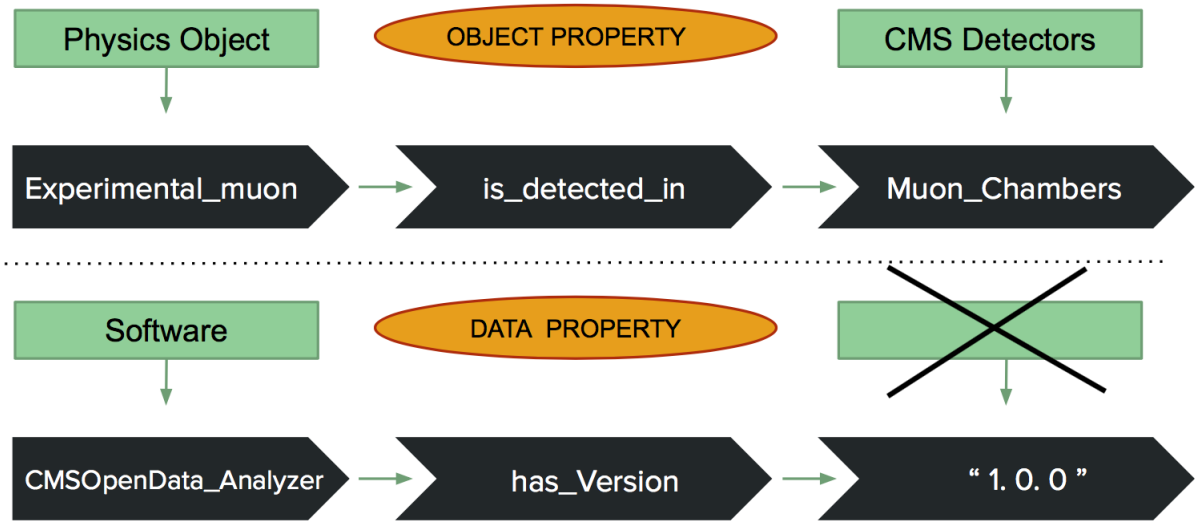
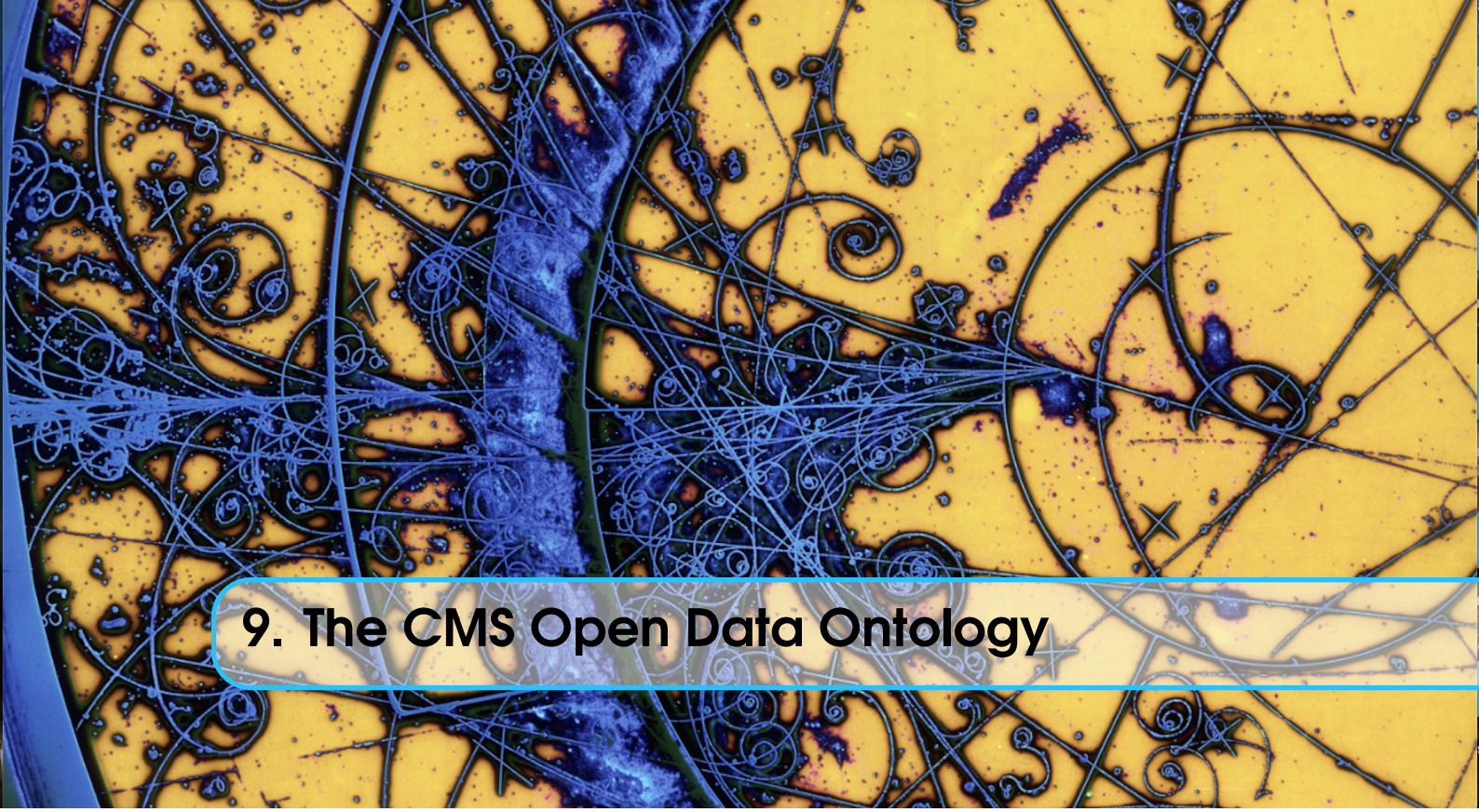


Figure 8.1: Object and Data properties diagram attached to ontology individuals.



9. The CMS Open Data Ontology

9.1 Download the code

The ontology has been written in RDF documents where informations in triplets are been save. The best way to access to this document is through the github repository at gcanasherrera/CmsOpenData_Ontology. You can clone this repository through git opening a terminal and typing the next command line,

```
git clone https://github.com/gcanasherrera/CmsOpenData_Ontology.git
```

In case you do not have git installed (an a error message appears when you type the line above-placed), or even you do not know what git is, please, go to Part II, chapter 3 (Requirements), section 3.2.

Remind that the repository will be downloaded in the same directory you are place at. For instances, if you are in *HOME* when you open the terminal and you perform the above command line, the repository will be downloaded at *HOME* (very obvious, but worth remind it). The ontology is just the **.rdf file**, and this is what you have to care about.

9.1.1 Ontology Structure

The Ontology was designed from top to bottom, meaning that first it class conceived to have the most important part that will have information about the High Energy Physics Process. The Ontology structure according to its main classes is presented in Figure 9.1. These parts are,

- **Standard Model**: includes information about the elementary particles, the fundamental forces and the interactions between them.

- **Events**: describes what the data type is and what the preservation policy is, the physics objects we are analysing and information about the vertex.

- **Analysis**: describes the CMS detectors, the particles we want to detect, the candidates particles from the decay and information about the track.

- **Software**: provides information about the software used for the analysis (authors, versions, required libraries and packages..)

- **Documentation**: used to register information not only about the software but the internal notes for the group as well as future

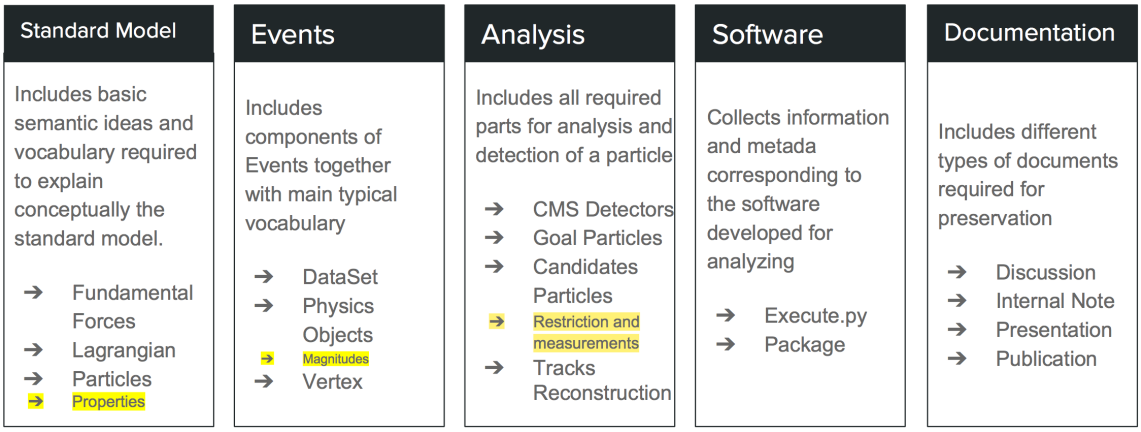


Figure 9.1: Ontology Class Diagram showing the main classes and their description and their subclasses.

9.2 Managing the CMS Open Data Ontology

As you can imagine, having just the .rdf document is not enough to extract information from it. In the last chapter, you were introduced to Protégé as the user-friendly tool for ontology work. Nevertheless, there are some other possibilities. You may use new software similar to Protégé or move the information from the .rdf file to a Database. A database is an organized collection of data. It is the collection of schemes, tables, queries, reports, views and other objects.

On the other hand, database management system (DBMS) is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Therefore, you can use a DBMS to manage the ontology and extract information from it. An example of a DBMS

9.2.1 In Protégé

To upload the .rdf document to Protégé, just open Protégé and click in File/Open... and browse until the directory you download using git clone and click at the .rdf file.

Once that you have uploaded the .rdf file you can obtain an idea of how the Ontology looks like clicking at diverse tabs. Protégé opens just due to the setup some tabs such as,

1. Active Ontology: include information about the ontology IRI, axioms, properties in the documents...
2. Entities: gives information about the class structure and hierarchy.
3. Individuals by class: shows the created individuals together with their usage.

4. Data Properties: shows the included data properties.
5. Object Properties: shows the included object properties.

Remind that these already-opened tabs can vary as a function of Protégé's version. In case you lack one of these tabs, you can open a new one just clicking at Window/Tabs. I strongly suggest taking your time navigating around the tabs looking at the different classes and individuals that the Ontology contains.

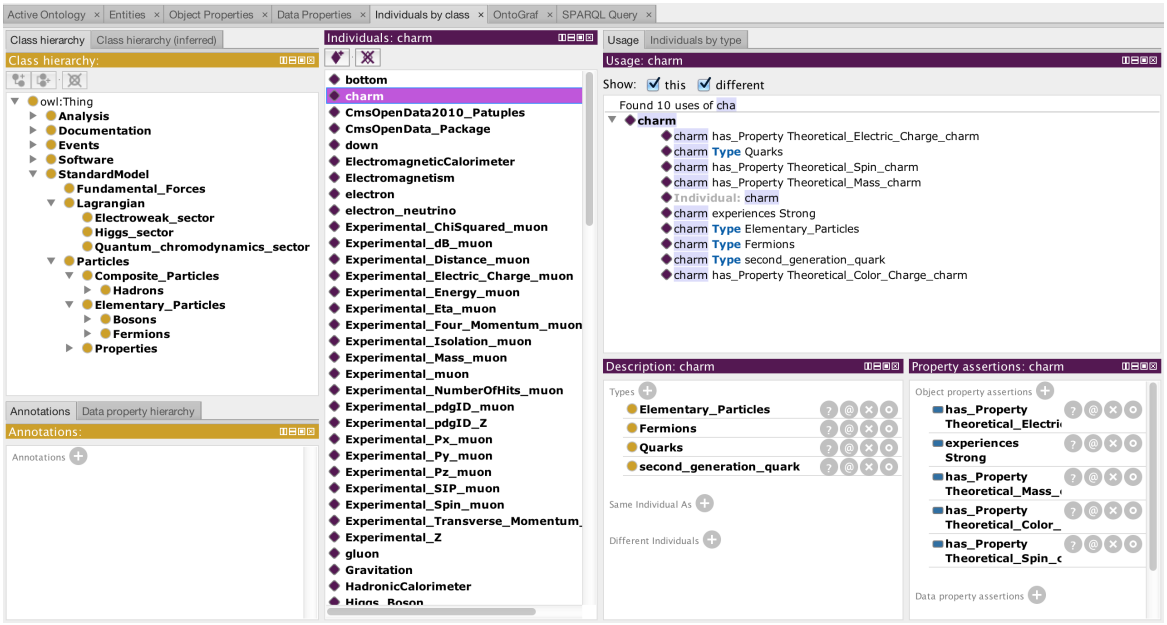


Figure 9.2: Screen Capture of Protege App at the individuals tab view for the High Energy Physics Ontology

One of the best features of Protégé is that it provides you the chance of creating a graph of the classes and subclasses together with the attached individuals (see Figure 9.3.). This view is obtained thanks to the OntoGraph Tab. In this tab, you only need to click over the different bubbles to expand their content. There is a way to identify quickly whether

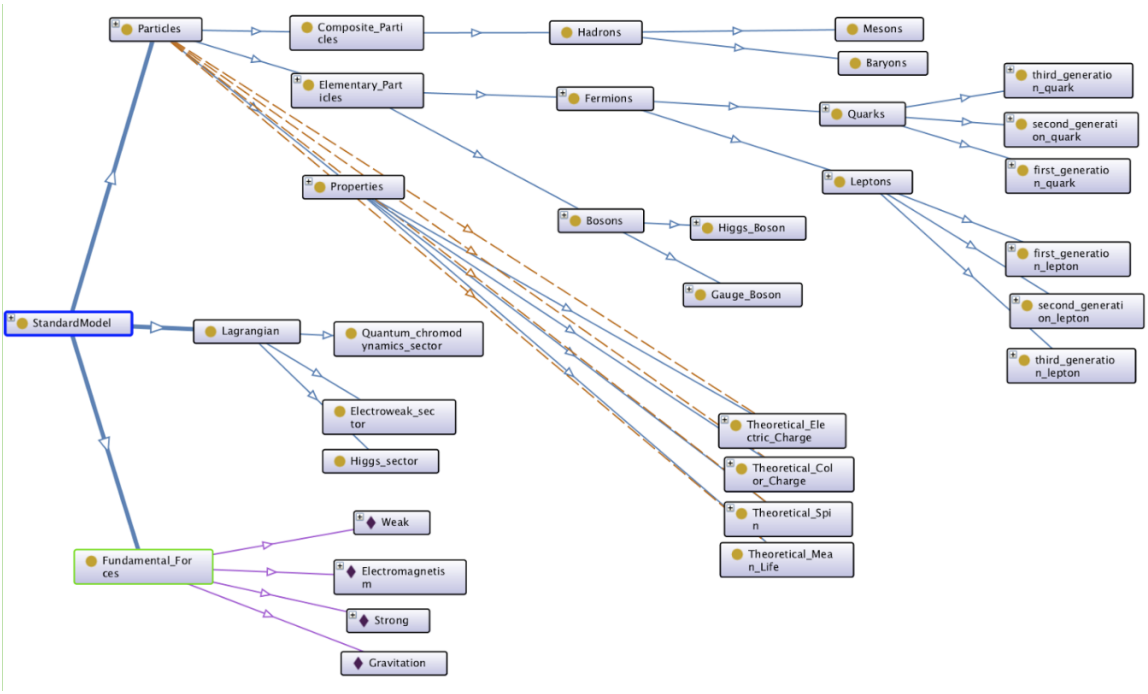


Figure 9.3: Screen Capture of Protege App at the OntoGraph tab view for the High Energy Physics Ontology's class *Standard Model*

9.2.2 Query the Ontology

The queries are written in SPARQL, a language designed to ask precisely RDF documents and obtain information from them. The used graphical interface is *Protégé*. In its last version, *Protégé* includes a TAB prepared for SPARQL queries. This is the tool used to get the results presented in this report.

9.3 Some SPARQL Use Cases Examples

9.3.1 Ask to the Standard Model Class

These queries have been posed to the Ontology main class based on the Standard Model

Example 1: Fermions types

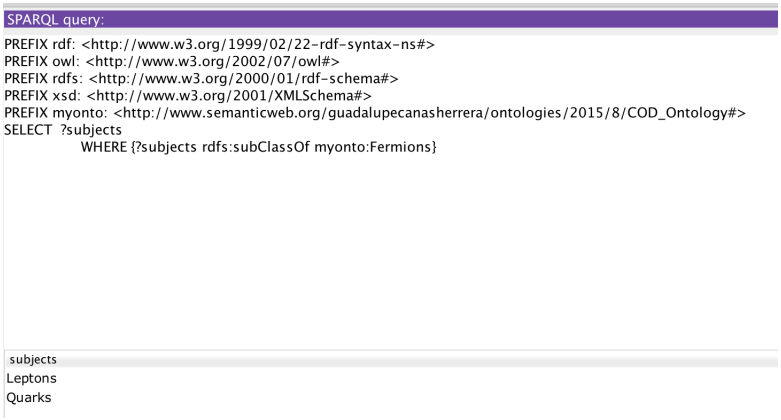


Figure 9.4: Example of SPARQL Query asking about Fermions types to Particles SubClass

Example 2: Quark types

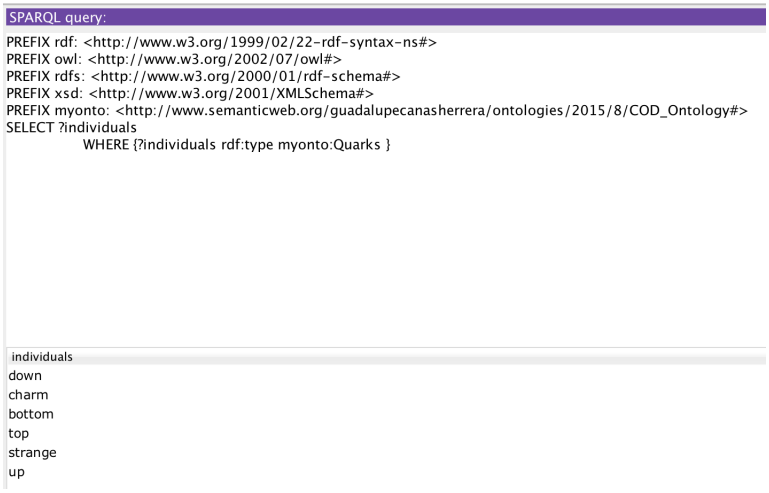


Figure 9.5: Example of SPARQL Query asking about different Quark types in the Standard Model

Example 3: Lepton types

```
SPARQL query:
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX myonto: <http://www.semanticweb.org/guadalupecanasherrera/ontologies/2015/8/COD_Ontology#>
SELECT ?individuals
        WHERE {?individuals rdf:type myonto:Leptons }
```

```
individuals
tau_neutrino
muon
Experimental_muon
electron_neutrino
muon_neutrino
electron
tau
```

Figure 9.6: Example of SPARQL Query asking about different Leptons types in the Standard Model

Example 4: Fundamental Forces

```
SPARQL query:
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX myonto: <http://www.semanticweb.org/guadalupecanasherrera/ontologies/2015/8/COD_Ontology#>
SELECT  ?forces
        WHERE {myonto:muon myonto:experiences ?forces}
```

```
forces
Weak
Electromagnetism
```

Figure 9.7: Example of SPARQL Query asking about what forces the muon experiences according to the Standard Model

Example 5: Theoretical properties

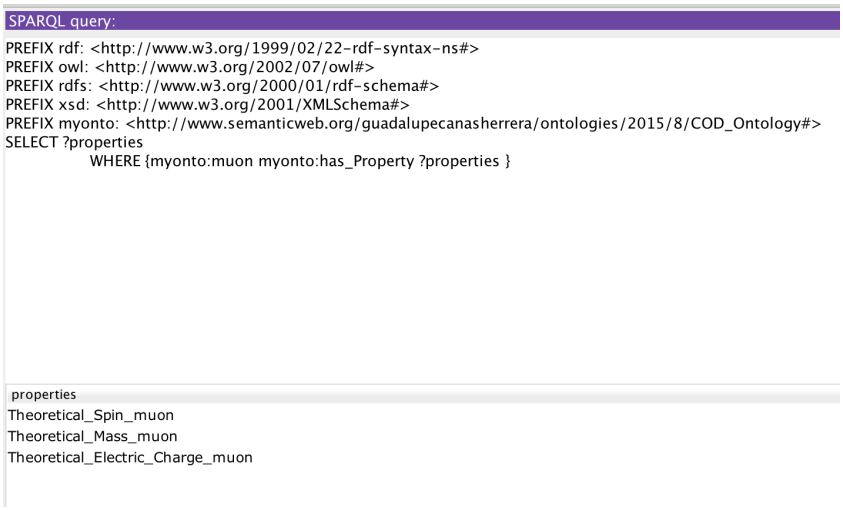


Figure 9.8: Example of SPARQL Query asking about what properties the muon has attached according to the Standard Model

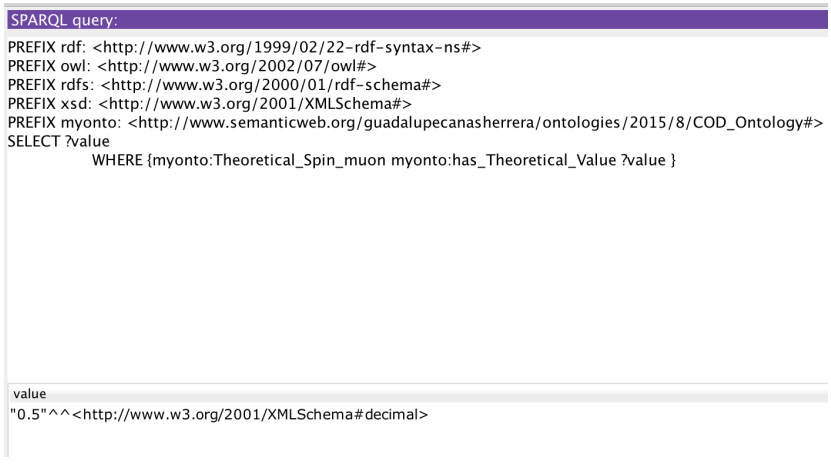


Figure 9.9: Example of SPARQL Query asking about a concrete value of a muon's property

Example 6: Filtering theoretical properties values

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX onto: <http://www.semanticweb.org/guadalupecanasherrera/ontologies/2015/8/COD_Ontology#>
SELECT ?individuals ?properties ?values
        WHERE { ?individuals rdf:type onto:Quarks .
                ?individuals onto:has_Property ?properties .
                ?properties onto:has_Theoretical_Value ?values
                FILTER (?values >= 4)
               }
```

individuals	properties	values
bottom	Theoretical_Mass_bottom	"4.18"^^<http://www.w3.org/2001/XMLSchema#decimal>
top	Theoretical_Mass_top	"173.34"^^<http://www.w3.org/2001/XMLSchema#decimal>

Figure 9.10: Example of SPARQL Query filtering possible Quarks' theoretical values.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX onto: <http://www.semanticweb.org/guadalupecanasherrera/ontologies/2015/8/COD_Ontology#>
SELECT ?individuals ?properties ?values
        WHERE { ?individuals rdf:type onto:Fermions .
                ?individuals onto:has_Property ?properties .
                ?properties onto:has_Theoretical_Value ?values
                FILTER (?values >= 1)
               }
```

individuals	properties	values
top	Theoretical_Mass_top	"173.34"^^<http://www.w3.org/2001/XMLSchema#decimal>
bottom	Theoretical_Mass_bottom	"4.18"^^<http://www.w3.org/2001/XMLSchema#decimal>
tau	Theoretical_Mass_tau	"1.77682"^^<http://www.w3.org/2001/XMLSchema#decimal>
charm	Theoretical_Mass_charm	"1.29"^^<http://www.w3.org/2001/XMLSchema#decimal>

Figure 9.11: Example of SPARQL Query filtering possible Fermions' theoretical values.

9.3.2 Ask to the Documentation Class

This query is related to the Documentation Class

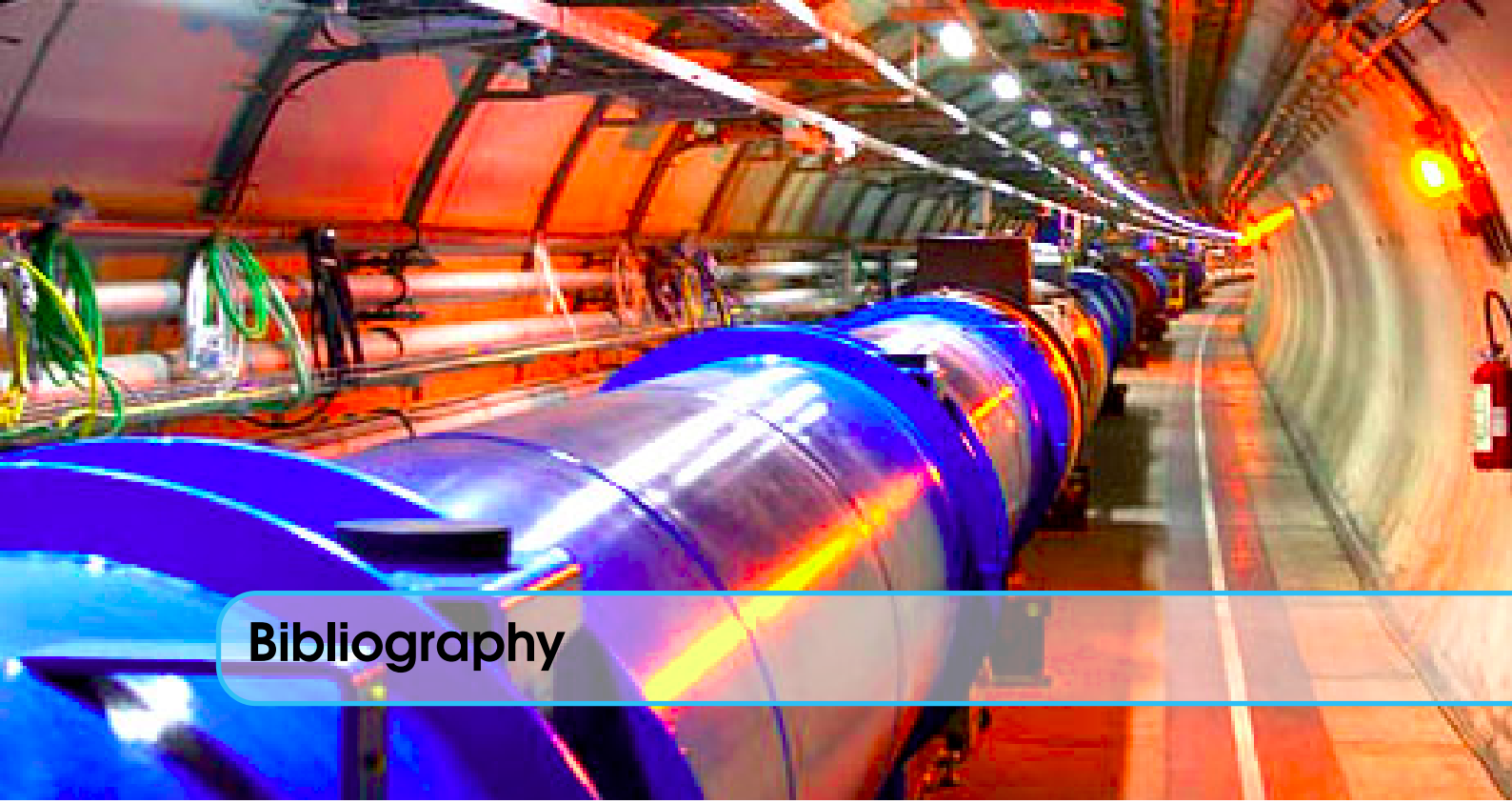
SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX onto: <http://www.semanticweb.org/guadalupecanasherrera/ontologies/2015/8/COD_Ontology#>
SELECT ?individuals ?name
        WHERE { ?individuals rdf:type onto:Software .
                ?individuals onto:has_Authors ?name .
                FILTER (regex(?name,'^Palmerina Gonzalez'))
              }
```

individuals	name
CmsOpenData_Package	"Palmerina Gonzalez"@

Figure 9.12: Example of SPARQL Query filtering possible Fermions' theoretical values.

9.3.3 In a Database



Bibliography

Books

Articles

https://www.clear.rice.edu/mech517/Books/oop3.pdf