Building a web application for the purpose of monitoring the Cherenkov Telescope Array telescopes and studying parameter correlations

Author: Ana Prpić

Supervisor: izv. prof. dr. sc. Dejan Vinković

Master's Thesis in Physics

Split, September 2015

Department of Physics
Faculty of Natural Sciences and Mathematics
University of Split

Table of Contents

List of Figures

List of Tables

Abstract

In this thesis, a single-page web application for the proposed Cherenkov Telescope Array (CTA) observatory was developed and reviewed in detail. A web application is the simplest, most elegant solution to the problem of open data access for teams that are not physically located at one place. All they need is a computer and internet connection to access the data and data visualizations. The CTA observatory has a plan of becoming an open observatory, with public access to the data, so a web application is a logical step in allowing open access from anywhere in the world. The entire application was built using open-source programs and libraries, using Python programming language for the back-end, and JavaScript for the front-end. The application is enabling users to visualise of telescope parameters and recorded data from the MAGIC telescopes. To visualize the data, user must select a date range, telescope, statistics and parameters of interest. Also, an analysis of the telescope parameters was done in order to try to find relationships between them. Methods used were ordinary linear regression, RANSAC and Theil-Sen. The analysis was conducted using Python libraries for data analysis, such as Pandas, NumPy, SciPy, Sci-Kit Learn and others.

Acknowledgements

I am taking this opportunity to give a very special thanks to my thesis advisor, dr. Dejan Vinković for his guidance and direction. His enthusiasm, support and patience truly know no limits!

I would also like to express my gratitude to prof. dr. Nikola Godinović, for introducing me to the idea of building a web application. It has proven to be a great challenge and am extremely thankful for the opportunity.

To my loving family, I give thanks for the unwavering love and support.

Finally, I would like to dedicate this thesis to my best friend and love, Dubravko Balić, who has always provided technical, moral and every possible kind of support!

Introduction

Significant breakthroughs in the fields of astronomy and astrophysics usually follow the invention and development of a new instrument. One such breakthrough was achieved in the year 1988 with the development of a 10 m diameter optical reflector called the Imaging Atmospheric Cherenkov Telescope (IACT) at the Whipple Observatory. This technological advancement led to the discovery of the first VHE gamma ray source, the Crab Nebula in the 1989. Soon, it has become evident that the study of the γ - rays is an essential tool in the study of the fundamental astrophysical phenomena, as well as cosmology and high - energy physics. [1]

Since then, hundreds of new gamma ray sources have been discovered, some of which are showing unexpected features and it has become obvious that a more detailed investigation needs to be conducted in order to get more and better data. The technical equipment needed for this has to have at least an order of magnitude better sensitivity, better energy coverage than present telescopes (they have the sensitivity of about 1% of the Crab nebula flux, energies being between 0.1 - 1 TeV), a better angular resolution (a typical angular resolution is 0.1° or slightly better for a single gamma ray, but sufficiently intense point sources can be located with a precision of 10 - 20 arc seconds [2]) and time resolution on at least sub-minute scale (current time scale is of a few minutes).

The Cherenkov Telescope Array (CTA), is a multinational, world-wide initiative to build a new generation ground-based γ-ray instrument [3] that proposes to solve the technical limitations of current instruments and achieve the desired performance goals.

Also, the proposed new way of recording and storing the recorded data presents us with the need for a specialized web application, one that would eventually serve for monitoring purposes and quality assurance by means of tracking telescope performance as well as providing a visual tool for displaying the recorded data. In this work such an application was built using open-source technologies like Python, PyMongo, Bottle etc.

The data science is an increasingly popular and growing field, with the development of modern technology where numerous applications are tracking everything we search and buy, how we sleep, where we go, how many steps we make in a day etc. This is an exciting field, used by many people and organizations to employ data-driven decisions. Now, it is easier than ever to gain access to data and extract valuable information from it, especially with Python programming language and its many accompanying libraries.

1 High-energy astrophysics

The advances in fields of astronomy, astrophysics and cosmology is driven by the development and implementation of new devices, instruments, techniques and technologies for observing the Universe. The advances made in the field of computer science and exponential growth of computer speeds were also an important factor in this revolution. The term 'high-energy 'astrophysics describes the processes involving high energy and how they relate to astrophysical and cosmological problems. An example of these processes is a charged particle accelerated to extremely high energies in an astronomical setting, e.g. in a binary system. This field of physics allows us to study processes involving energies beyond those attainable in accelerators back on Earth in order to either use observations to confirm existing laws of physics or discover possible new ones.

1.1 The VHE gamma-rays

The term 'gamma rays' is a broad term that refers to photons of electromagnetic radiation that have very short wavelength and, therefore, very large energy. These photons have energies from 100 keV (below that are the X-rays) up to 100 TeV. Note here that this is about 15 orders of magnitude, so this energy span is larger than the rest of the entire known electromagnetic spectrum!

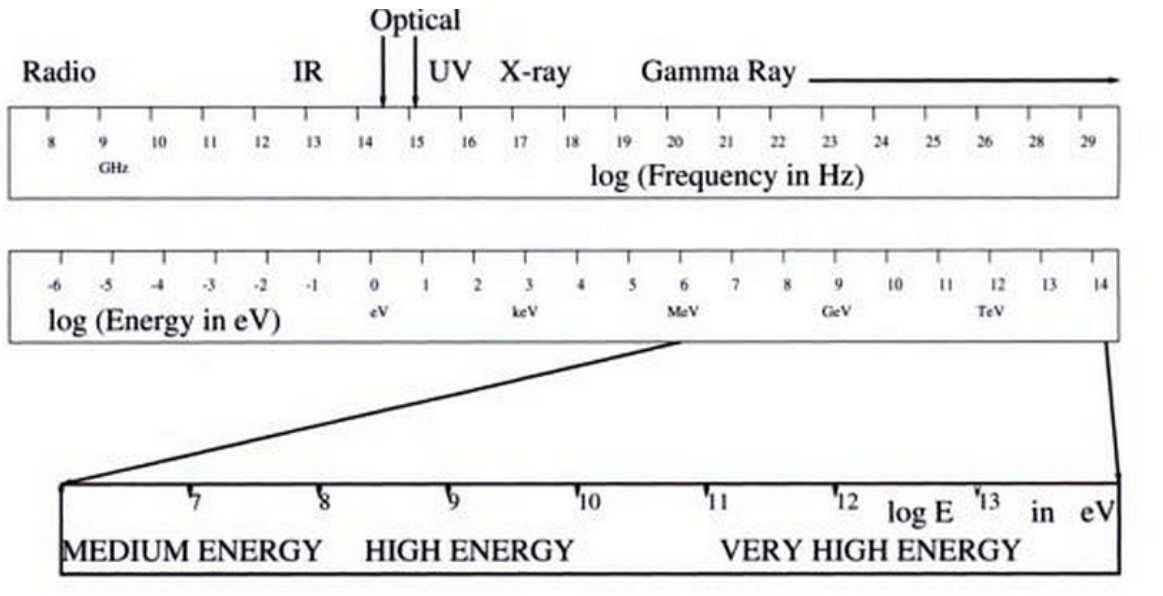

Figure 1.1: Electromagnetic spectrum showing the full extent of the spectrum part covered by the generic term gamma rays (Source: T.C.Weeks: Very High Energy Gamma- Ray Astronomy, page 4)

Therefore, the detection methods for this huge span vary and are divided into several bands. The band of our interest here is the one of the so-called very high energy spectrum.

The very high energy (VHE) band is roughly the area between 100 GeV to 100 TeV. Important thing to note here is that these values are not defined by some physical process but by techniques used in their detection and how they interact.[4] The VHE detection relies on the particle cascades produced by the gamma ray interacting with particles in our Earth's atmosphere. Gamma rays entering the atmosphere interact with air molecules and dissipate their energy by producing electron-positron pairs, which then dissipate theirs by radiating photons via bremsstrahlung process. This leads to almost exponential growth in the number of particles as they go deeper into the atmosphere and lose energy. We call this process a cascade and the bundle of particles an air shower. Once the energy of particles is less than $\sim$ 80 MeV, the shower electrons are unable to produce a secondary ray so at this point the number of particles decreases with increasing depth. The particles are at their maximum $\sim$ 10 km above the ground for a 1 TeV gamma ray.

1.2 Cherenkov radiation

When the secondary charged particles are travelling through the atmosphere, they are disrupting the local electromagnetic field in the air and are temporarily displace and polarize electrons in the atoms. When the disruption passes, the equilibrium is restored by photon emission. If the secondary particles are travelling faster than the phase velocity of these photons in the air, they emit optical radiation; a light shock wave. This is manifested as a brief (duration of about $\sim$ 5 ns) flash of blue light called Cherenkov Radiation. This light is then detected by ground-based telescope arrays. The total area of the ground that is illuminated by this flash is $\sim$ 10 000 m^2.

The characteristic distance scale for these interactions is called the *radiation length* and is defined as the distance over which an electron loses most of its energy by bremsstrahlung. The atmosphere is thought to have around 30 radiation lengths of material but ground-based telescopes only record a fraction of the total radiation and particle content of the shower.[7] (Fig. 1.2)

At the energies considered here, gamma rays are generated by cosmic-ray particles - most often by relativistic photons or nuclei colliding with nuclei in atmospheric gas, or by electrons up-scattering less energetic photons.[23] However, more generally speaking, cosmic rays cause hadronic cascades and therefore have a wider angular distribution than the gamma-ray showers, which cause leptonic cascades and have a much smaller angular distribution. (Fig. 1.3)

The cosmic rays produce 99% of the air showers while gamma rays produce only 1%, hence the Cherenkov light flashes that are due to the gamma rays are harder to detect.

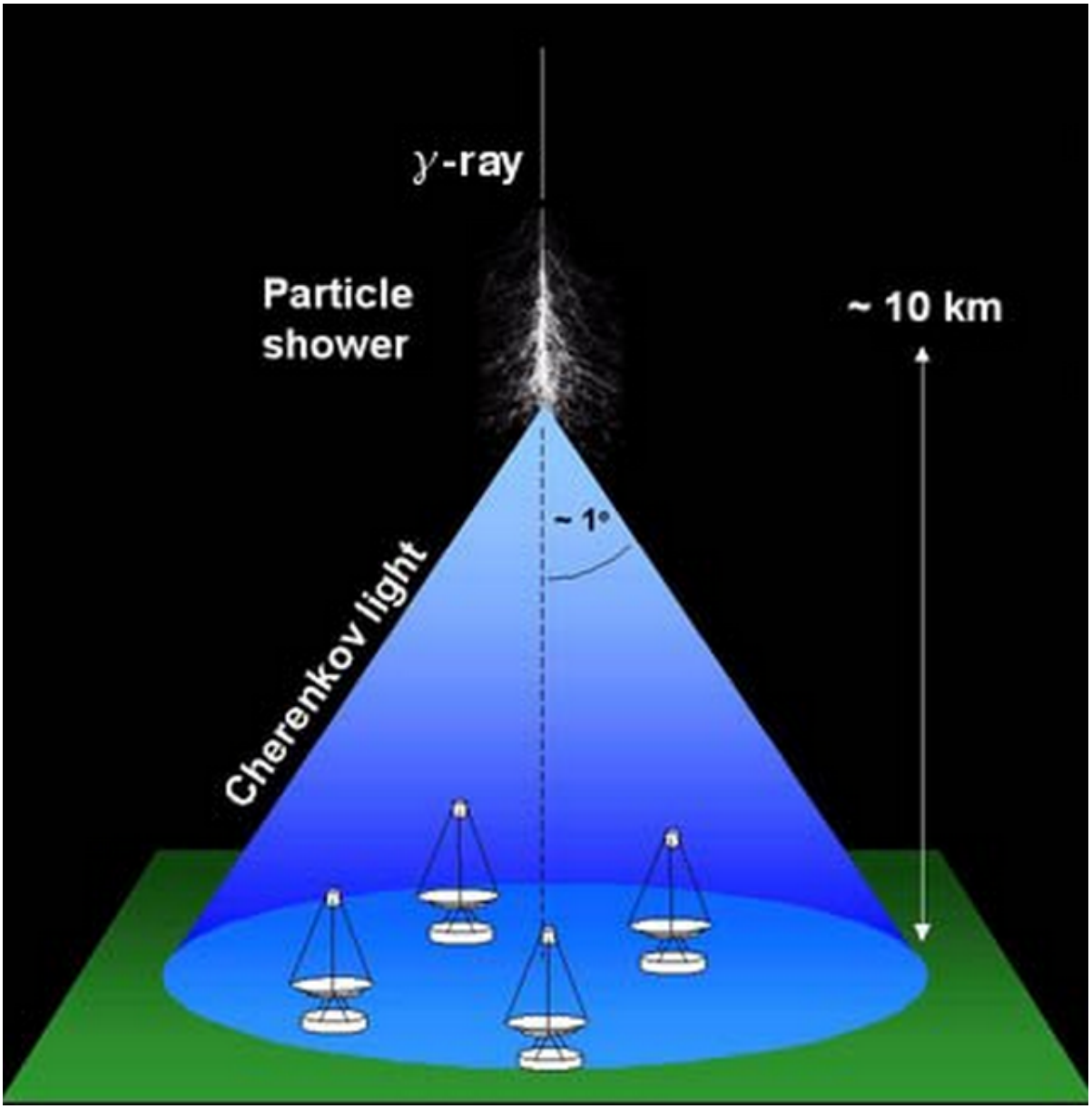

Figure 1.2: A schematic of ground-based gamma-ray astronomy (From: [5])

1.3 Why the gamma rays?

The very high-energy photons of electromagnetic radiation travel towards the Earth from different sources inside our Milky Way Galaxy and beyond, carrying with them unique pieces of information about most violent and energetic physical processes in our Universe. This is possible because gamma rays are unaffected by electric and magnetic fields in space, so they can travel great distances without being absorbed by intergalactic dust or gas and are showing us exactly where the rays are coming from. We use them to find out more about distant regions of the Universe or otherwise obscured parts of our own Galaxy.

The gamma rays are interesting for studying because they provide not only information

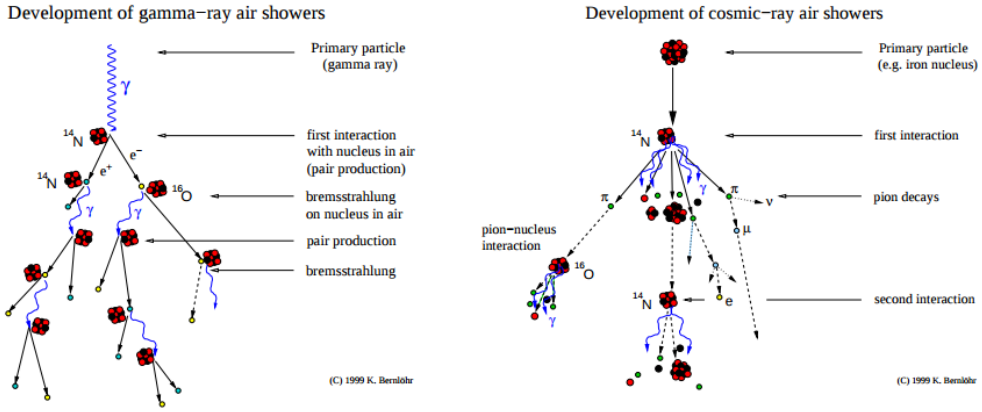

Figure 1.3: A schema of the air shower development (From: [6])

about their sources, which act like giant particle accelerators in Space, but can also shed some light on the origin of the cosmic rays. They are arriving at the Earth from all directions and it is hard to calculate where exactly they came from because their direction is changed due to the magnetic field of the galaxy - they do not point back to the place of their origin. The general consensus is that the place of origin for cosmic rays is also the place where gamma rays originate from. The established sources for the gamma-ray radiation are supernova remnants (SNRs), Pulsar Wind Nebulae (PWNs), gamma-ray bursts (GRBs), micro quasars and binary systems (containing either two massive stars, a white dwarf (WD), a pulsar, or a solar mass black hole). Other potential sources inside the Galaxy are isolated massive stars and white dwarfs, or accreting neutron stars and white dwarfs. Outside the galaxy, an active galactic nuclei (AGNs) and radio galaxies (e.g. M87) are potential sources.

Also, the VHE gamma rays can be used to probe physics beyond the Standard Model and search for dark matter signals or even quantum gravity. Possible targets include the Galactic center, nearby dwarf galaxies and selected extragalactic sources.[8]

2 The Imaging Atmospheric Cherenkov Telescopes

The Cherenkov telescopes are currently the most efficient ground-based detectors of high energy gamma-ray particles.[10] They use mirrors to focus the Cherenkov light onto photon detectors. For near vertical showers the Cherenkov light illuminates a circle with a diameter of about 250 m on the ground. This light can be captured with optical elements and be used to image the shower. Reconstructing the shower axis in space and tracing it back onto the sky allows the celestial origin of the gamma-ray to be determined.[11] (Fig.2.1)

As any other optical or radio telescope, an IACT consists of three basic elements: a mechanical tracking system, which is tracking objects on the celestial sphere to compensate for the

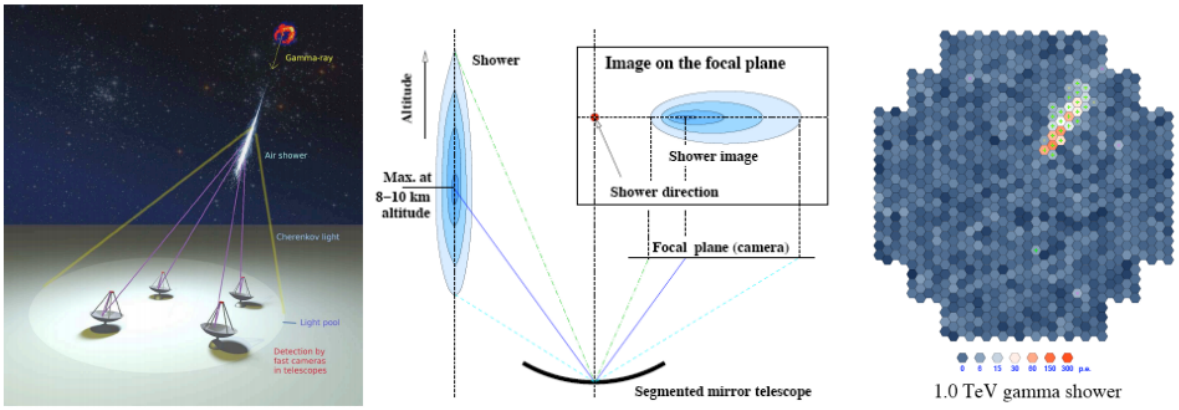

Figure 2.1: Left: Schematic of the Cherenkov light pool, originating from a primary γ-ray and illuminating an array of telescopes. **Central:** Shower imaged by a telescope. The shower image in the camera has an elliptical shape and the shower direction lays on the extension of its major axis; the image intensity is related to the primary energy. **Right:** Image of a γ-induced air shower in the camera. (From:[9])

Earth's rotation, a collecting surface, which gathers the incident electromagnetic radiation and focuses it, and a receiver element, which converts the collected light into a recordable image of the observed field of view (FOV).[12] Being optical instruments, they can operate only on moonless and clear nights.

As the Cherenkov light is fairly faint, a very sensitive light detectors are used for their detection: the PMTs – Photomultiplier Tubes. (Fig.2.2) They consist of a photocathode, an anode and a couple of dynodes. A dynode is an electrode in a vacuum tube that upon interaction with an electron beam through the process of secondary emission is emitting a large number of electrons. As the initial photons of the gamma ray arrive, they create a group of primary electrons that are then accelerated towards the first dynode by an electric field. These dynodes are organized in such a way that each one is on a slightly more positive potential than the one before, so that each dynode attracts the electrons from the previous one and is releasing even more electrons to the one coming next. This causes a cascade that increases the number of electrons that are produced inside the tube exponentially! In present Cherenkov telescopes the typical focal camera is constituted by a set of single PMTs packed together and each one is constituting a single pixel.[12]

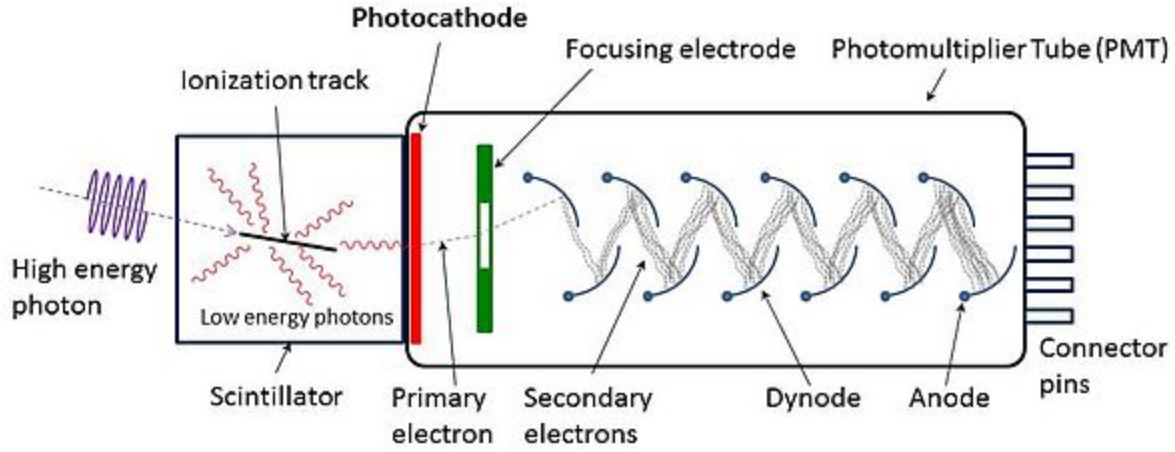

Figure 2.2: An illustration of a photomultiplier tube (From: Wikipedia)

The image formed by the camera is a projection of the shower. Cherenkov photons emitted at different heights reach the mirror dish with different angles and are focused on different positions in the camera of the telescope. As a consequence, the image contains information of the longitudinal development of the EAS, i.e., the number of particles emitting Cherenkov light as a function of the height in the atmosphere. Light coming from the upper part of the shower, where the secondary particles are more energetic, has smaller Cherenkov angles and is mapped onto a region close to the camera center, whereas light emitted from the last stages of the shower has larger Cherenkov angles and is mapped further away from the camera center.[15] (Fig.2.3)

The Cherenkov telescopes have an important ability: they discriminate against events generated by cosmic rays. Also, they measure the total light generated by the shower, which is proportional to the energy of the primary gamma ray.

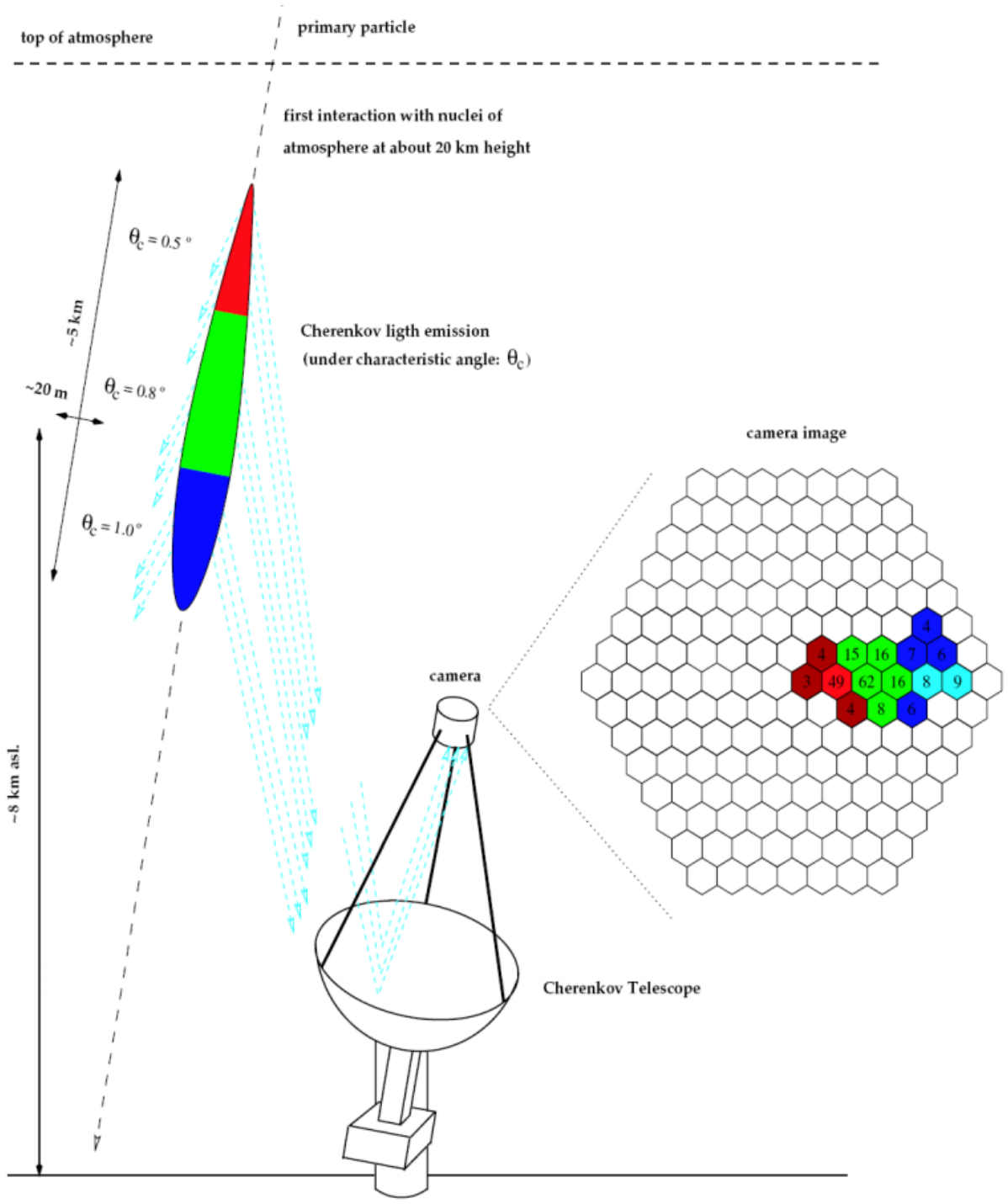

Figure 2.3: A sketch of the principle behind image formation. (From: www.columbia.edu)

The current generation of IACTs are the MAGIC and VERITAS in the northern hemisphere and HESS and CANGAROO in the southern hemisphere. (Fig.2.4)

Figure 2.4: The current generation of IACTs. (From: [12])

2.1 MAGIC

The Major Atmospheric Gamma Imaging Cherenkov (MAGIC) Telescope was designed in 1998.[16] with the main goal of being the IACT with the lowest possible gamma-energy threshold. The threshold energy is actually the minimum energy that a gamma ray has to have to induce the cascade and be detected by the telescope.

MAGIC is mostly based on the experience acquired with the first generation of Cherenkov telescopes coupled with a large number of technological improvements.[17] The initial motivation to head for the low-energy threshold was that there was already present a well populated sky map of sources detected around 10 GeV, but more than half of them were unidentified due to the poor angular resolution. The new detectors provided much larger effective areas (around

4 $\times$104 m^2) than previous, better angular resolution (ranging from 0.2 ° close to the threshold down to 0.1 ° at higher energies), acceptable energy resolution (going down from 30 % at the threshold energy to less than 20 % above 100 GeV) and a well-tested capability to separate gamma-rays from other backgrounds.

Figure 2.5: The MAGIC telescopes

The telescope itself is a large and light-weight Cherenkov telescope, which incorporates a large number of technical innovations and is located at the Roque de los Muchachos Observatory on the island of La Palma.[21] (Fig.2.5)

At the time of building the telescope, in the year 2003., the telescope's 17 m diameter reflector made it the largest IACT in the world. The telescope is composed of 964 pieces of 1 m^2 mirror panels amounting to $\approx$ 320 m^2 of total reflector area and a 577 photomultiplier-tube camera. The mirrors are fully steerable using an active mirror control system in order to better reposition to a certain position on the celestial sphere after receiving an alert about a GRB from a satellite.

The overall reflector shape is parabolic to minimize the time spread of the Cherenkov light flashes in the camera plane. The preservation of the time structure of the Cherenkov pulses is important to increase the signal-to-noise ratio with respect to the night-sky background light (NSB).[22]

Soon after, in the 2005, the construction of a second telescope was initiated, 85m away from the first one. The second telescope was mechanically identical, but some new features were added that have helped in stronger suppressing of background light, a stronger distinction for hadronic-based showers and lowering the energy threshold (around 30 GeV). Also, the stereoscopic view provided the ability of tracking the cascades from two different perspectives which allowed increased precision in locating the direction of arrival of each gamma ray using the intersection of the axes of the two images.[23] After the data are collected, gamma ray candidates are selected based on the shape of the shower images. The projection of shower

images on the focal plane result in ellipses with their major axes pointing in the direction of the source. The elongation of the shower image increases with increasing angular distance between the image centroid and the source position, allowing the reconstruction of the point of origin for each detected gamma ray with a single telescope. [7]

Among the most outstanding results obtained by TeV astronomy is the recent discovery of pulsed γ-ray photons above 25 GeV from the Crab Pulsar, which was not expected.[24] This is an important result that is providing a unique insight into the structure of pulsar magnetospheres and the main energy transfer processes at work.[12]

Weather and atmospheric conditions monitoring

The MAGIC telescope produces $\approx$ 70 GB/h of data each night (it iss $\approx$ 500 - 600 GB of data in only one night!). Additional data from the telescope control system, weather information station and the position of the camera are recorded as well. The continuous monitoring of weather and atmospheric conditions is necessary because extreme weather conditions can damage the telescopes and need to be monitored for safety reasons and because the presence of aerosols, dust and clouds in the atmosphere affects the transmission of the Cherenkov radiation. These conditions affect the data quality and telesope's performance. For example, the presence of clouds may reduce the trigger rate or when aerosols are present near the ground, there is a reduction in the number of photons which affect the energy reconstruction. MAGIC is triggered when several pixels in the camera exceed the threshold, within a coincidence time window of $\sim$ 2-25 ns.[20] The trigger system of the MAGIC telescope is a two-level system where the first level (L1T) applies tight time coincidence and simple next neighbour logic. The trigger is active in 19 hexagonal overlapping regions of 36 pixels each, to cover 325 of the inner pixels of the camera. The second level (L2T) can be used to perform a rough analysis and apply topological constraints on the event images. [21] The device that monitors the sky temperature as a measure of the presence of clouds in the atmosphere is the **pyrometer** (Fig.2.6). The pyrometer is mounted attached to the reflector surface of the MAGIC I telescope, and points to the same sky region as the telescope It measures the integral thermal radiation from the sky in the line of sight by a thermoelectric sensor in a wavelength range from 8 to 14 μm. An empirically derived function ('cloudiness') of the sky temperature, observational direction, air temperature and humidity reflects the presence of clouds. The 'cloudiness' is converted to a unitless value between 0 and 100. [19] All the data recorded are saved in an SQL database.

Figure 2.6: The radiation pyrometer and its installation attached to the MAGIC I reflector surface (From: [19])

3 Introduction to Cherenkov Telescope Array

3.1 Motivation and science

The current generation of ground-based Cherenkov telescopes (H.E.S.S., MAGIC, CANGAROO and VERITAS) have in recent years enabled imaging, spectroscopy and photometry of the gamma rays and have shown that the VHE gamma- ray astronomy may be entering its 'golden age'. Consequently, a new generation of VHE gamma- ray instruments is proposed in order to address the increasing needs of the scientific community. More specifically, to significantly improve the sensitivity, the observed energy band, the field of view, the signal sampling and to reduce the observing time.[12] This will give scientists the opportunity to explore the nature of black hole accelerators, properties of dark matter and quantum gravity.

Detectors of VHE photons need to have a large detection area due to the very low flux of the incoming photons; the effective area should be the size of at least the area corresponding to the Cherenkov light pool on the ground, which is about 105 m^2.

The Cherenkov Telescope Array (CTA) is conceived to allow both detection and in-depth study of large samples of known source types, and to explore a wide range of classes of suspected gamma-ray emitters beyond the sensitivity of current instruments, CTA will be a combination of the well proven technology of Cherenkov telescopes and of new wide-field gamma detectors. [12]

There are some core themes aimed for exploration with the CTA. The first one is understanding the nature and place of origin for cosmic rays and how particles accelerated in the shocks of supernova explosions to huge velocities impact their neighbourhood through interaction.The second one concerns understanding the nature and variety of particle acceleration around super-massive and smaller black holes. The objects of interest here are AGNs, GRBs, radio galaxies, μ-quasars etc. The third one is about searching for dark matter and physics beyond the Standard Model, in our own center of the galaxy. Dark matter probably exists in a form of an undiscovered particle class, as one of the most promising sources for dark matter is its annihilation radiation due to the predicted very high dark matter density. [3]

3.2 Performance goals

The CTA will consist of two arrays of Cherenkov telescopes in two different hemispheres, allowing the full coverage of the sky.[25] The arrays aim to: (a) increase the sensitivity by a factor of 10 for observations between 100 GeV to several TeV, which will be achieved by a combination of different-size telescopes: large, medium and small ones, (b) boost significantly the detection area and detection rates, which are particularly important for transient phenomena at the highest energies, (c) increase the angular and temporal resolution and, consequentially, the ability to resolve the morphology of extended sources and resolve flaring and time-variable emission on sub-minute time scales, (d) provide uniform energy coverage for photons from some tens of GeV to beyond 100 TeV, and (e) enhance the sky survey capability, monitoring capability and flexibility of operation. CTA will be operated as a proposal-driven open observatory, with a Science Data Centre providing transparent access to data, analysis tools and user training.[11]

The CTA presents a data organization and management challenge as well, not just technological. The southern telescope array will be producing around 0.4 GB/s of data and the northern around 1 GB/s – this amounts to roughly 25 petabytes of raw data in a year! For this reason, it is necessary to have a way of monitoring the telescope parameters while recording, to make sure all telescope parts are functioning within acceptable margins as well as an access to the recorded data. A web application would serve as a way of visualising the long-term MAGIC data but can be relatively easy adapted for the CTA telescope monitoring purposes.

The web application

1 Introduction

A web application is a computer program that enables visitors of a website to communicate with the data stored in a remote database, in order to retrieve the data from the database or save to the database, over the Internet, using a web browser. The data are then presented to the visitor inside their browser as information that is generated dynamically by the web application with the help of a web server. A web application is convenient because it is accessible to anyone who has an internet connection and because the application serves its purpose irrespective of the operating system and browser the client is using.

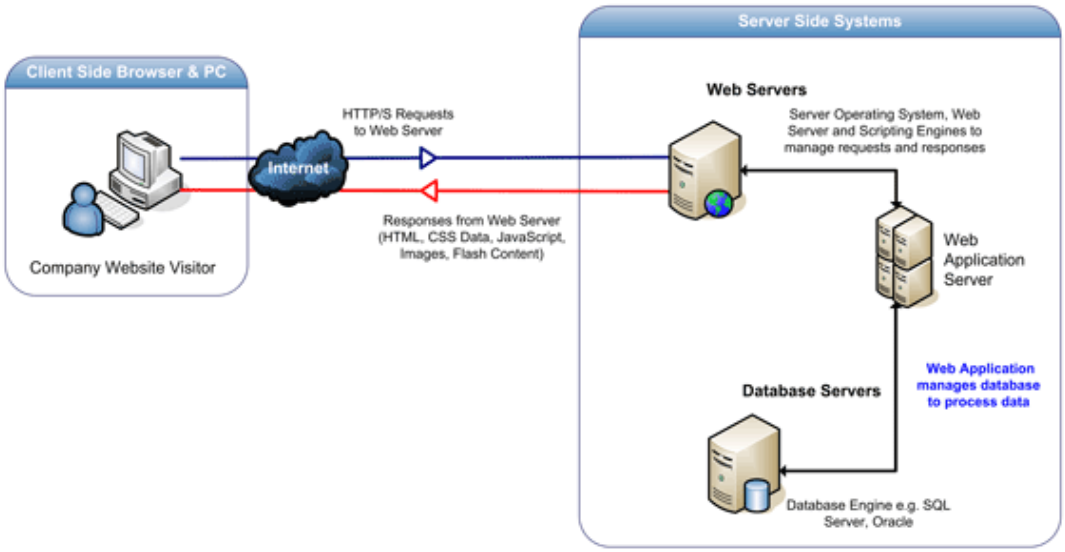

Figure 1.1: How web applications work (From: [26])

The figure 1.1 portrays a typical application that is made of three layers. The first layer is a web browser, the second layer is the dynamic content generation technology tool (i.e. Bottle framework) and the third layer is the database containing content (i.e. data recorded by the telescopes).

2 Why the web application

The web application can be used for parameter monitoring and correlation studying, data visualization and downloading data for further analysis. The user can select a date range, parameters of interest, the desired telescope and various data visualization options. The graphs are showing the selected parameter change in the selected time range (be it one parameter from two telescopes or two parameters from the same telescope). The option to plot two parameters on the same graph helps in spotting possible parameter correlations. In the future, the queries should be not only accepting dates but also times, which could prove to be very useful in early problems detection and telescope monitoring. This was not possible to implement right away, as the given data set does not have timestamps (because the values are calculated static averages of the data collected through the course of one night, and not the raw recorded data).

3 SQL vs. NoSQL

We use the term NoSQL (Not Only SQL) to describe an approach to managing huge amounts of data and database design, useful for the sets of distributed and unstructured data. The name describes a variety of different technologies that were developed as a response to the explosion of internet usage and the use of mobile devices which led to the need for storing increasing amount of user-related data with frequent queries to the database, increasing processing speed and big-data performance issues. They usually sacrifice consistency in favour of availability and speed. Relational databases were not designed to cope with the challenges that modern applications face.

The term SQL (Structured Query Language) usually describes relational databases that store relational data, which means that data in table A can have direct relation to information stored in the table B. The SQL stores data as rows in tables, similar to the spreadsheet concept. Since different data types are stored in different tables, they are joined when queries to the database are executed. The NoSQL stores data in a similar way, but with two columns only: the key and the value one. The identity **key** of the data record is not part of the object itself, which allows for distribution because the key can be used to identify the node where the object is physically stored. The **value** can be a collection of other objects, i.e. array or list, which allows easy storing of complex documents.

Relational databases require that you define a schema for storing the data beforehand. This approach does not agree well with the Agile development philosophy, because with each agile sprint adding new functionality possibly means changes to the original schema of the database. This requires migration of the old database to the new schema, which involves database downtime. Agile methodology is an approach to project management used in software development. The development is divided into two-week cycles, known as 'sprints' at the end of which teams

release smaller pieces of software, immediately usable. We can see that in this highly dynamical setting, a database downtime is not an option.

The relational databases require the entire base to be stored on a single server to ensure reliability and accessibility of the data (there are ways to bypass this, but generally this is not an intrinsic property of the SQL databases) while the NoSQL databases support saving data across multiple servers. So, the NoSQL databases scale horizontally by supporting distributed computing (which increases performance) and SQL only scales vertically within a single server, by increasing RAM memory or using faster disks, etc.[27] However, strict rules governing the relational database structure ensures data security and integrity.

So, when does one need to use a NoSQL database? There are several cases when this is recommended:

- when you expect a high writing frequency to the database as the NoSQL supports fast loading huge amount of data to the database
- when you know that in the very near future you will have to store huge amount of data, because the SQL database's performance degrades significantly when the amount of data in one table is larger than 5 GB
- when your data is location-based and you need to find data from specific locations, fast.

4 The technologies used

All the technologies selected for the making of this application are open-source technologies. What that means is that either the pieces of software are free for personal use via a free license or their source code is available for use without restrictions. The latter is interesting because of the possibility for anyone to contribute to the development of such software. The technologies are as follows:

- A Scientific Linux server machine, where the database and the web application are hosted,
- MongoDB database, a data storage,
- Bottle, a micro-framework used to relay queries received from the web-page to the database,
- PyMongo, a library used for interacting with the database from a Python application (the package bridges the gap between Bottle and MongoDB)
- Python programming language
- dygraphs, a fast and flexible open source JavaScript library for data visualization purposes,
- Bootstrap, a front-end web development framework that is a collection of HTML/CSS/-JavaScript components and
- jQuery UI, a set of widgets and user interface interactions build on top of the jQuery JavaScript library.[32]

All of the programs mentioned were installed using the Linux command line, or shell that provides an interface between the user and the internals of a computer. It can be used opening a terminal emulator on a Linux computer. The shell is a programming language that is run by the terminal. The *bash* is the most used shell, that I also used for automating the process of starting the magic_api.py service. The used data come from the MAGIC Telescopes and contain static averages of all the parameters that are measured through the course of one night. The MAGIC project uses an SQL database so the data are exported from that SQL base to a .csv file and imported into MongoDB.

4.1 MongoDB

Concepts: Database, Collection, Document

What makes MongoDB special is that it stores data in rich structures (arrays of arrays and lists that eventually boil down to integers, floats, dates and strings). The basic way of communicating with the database is writing a query, passing it and getting back a cursor containing all the desired documents, over which we can iterate. MongoDB is a server process that runs on all major operating systems, such as Linux, Windows and OS X. The *mongod* process listens on port 27017 by default and clients can connect to it and insert, delete, update records and query through them. The MongoDB stores the data in files (the default location is /data/db/).

Database: MongoDB is a database that is essentially a set of collections. For example, a database for a blogging application we could name *blog* and it would normally have the following collections: articles, authors, comments, categories etc.

Collection: A collection exists as an entity in the database and we can visualise it as a folder. In MongoDB, collections do not enforce a schema as the database itself is schema-less and the documents within the collection are allowed to have different types of fields. Typically, all documents in a collection are arranged so that they share a related or similar purpose.

Document: A document is a record in a MongoDB collection and the basic unit of data in the MongoDB database. Documents stored in the database are analogous to JSON objects but within the database they exist in a more type-rich format known as BSON. A document typically contains a set of fields or key-value pairs. The best way to visualize what a document is, is to think of it as a multidimensional array that consists of key-value pairs. The document structure usually looks like this:

```
{
        field1: value1,
        field2: value2,
        field3: value3,
             ...
```

```
        fieldN: valueN
}
```

The value of a field_i can be any of the BSON data types, which means that the value_i can be documents, arrays, arrays of arrays, arrays of documents etc.

Installing and using MongoDB

To install MongoDB on a Ubuntu Linux machine, first create a repository file in the directory /etc/apt/sources.list.d/mongodb.list with the following command:

```
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart
        dist 10gen' > /etc/apt/sources.list./mongodb.list
```

When this is done, the repository cache needs to be refreshed:

```
apt-get update
```

Now we can install the MongoDB:

```
apt-get install mongodb-10gen
```

To start the MongoDB shell, we type the following command

```
mongo
```

into the command line. When the MongoDB shell is started, the following text appears on the screen:

```
MongoDB shell version: 2.4.9
connecting to: test
>
```

The data import into the MongoDB

As previously mentioned, the data we want to import into the MongoDB are actually data the two MAGIC telescopes record every 10 seconds through the course of one night, and save them to an SQL database. When the recording phase is finished, the data is processed, *rms, mean, median, min, max* values of the data recorded in that night are calculated and written to the SQL database. From there, the data is dumped into a .csv file. I imported the data from the .csv file into the MongoDB entering the following command in the terminal (**N.B. not** in the mongo shell, but in the server command line):

```
mongoimport -- db magicdb -- collection  collection_name -- type csv
        -- headerline -- file /path/to/file.csv
```

(The *mongoimport* utility connects to the *mongod* instance running on the local server machine on port number 27017.)
- - db magicdb - specifying the wanted name of our database in MongoDB
- - collection collection_name - specifying the name for the collection inside the base mongodb
- - type csv - specifying the type of the file from which the import is being made
- - headerline - titles from the .csv are imported to MongoDB as names for the parameters
- - file /path/to/file.csv - specifying the path to the .csv file from which the data is being imported

If we want to see what databases are available for use, we need to type the following command:

```
> show dbs
```

The following is shown:

```
local          0.078125GB
magicdb        0.203125GB
```

To use the database of interest, the *magicdb*, we type the command:

```
> use magicdb
```

The mongo responds with:

```
switched to db magicdb
```

If we want to see what is inside the *magicdb* database, we type:

```
> show tables
```

and the system responds with:

```
dataset_params
datasets
system.indexes
```

We can see that we have three collections. The system.indexes collection lists all the indexes in the database. The *dataset_params* collection contains the parameters used for data visualization: date, rms, mean, median, min and max.

The document structure

In the mongo shell, the primary method for the read operation is the db.collection.find() method. This method queries a collection and returns a cursor to the returning documents [28]. Our MongoDB documents are composed of key-value pairs and have the following structure:

```
{
        "_id" : ObjectId("55c91da306b6bb003a33e23f"),
        "id" : 20,
        "date" : ISODate("2012-11-19T00:00:00Z"),
        "name" : "camdt_aq",
        "mean" : 75.356774,
        "median" : 74.521135,
        "rms" : 8.979812,
        "min" : 53.833284,
        "max" : 173.484717,
        "telescope" : "M1"
}
```

We can see that we have two ids; one '_id 'and the other 'id'. The first one is assigned to every object upon creation or insertion into the database, and we can think of the *ObjectId* as of a globally unique identifier. The complex creation pattern of *ObjectId* makes all the _id fields unique. The *ObjectId* is a 12-byte BSON type and is composed of: the number of seconds that has passed since the unix epoch (first 4 bytes), the machine (on which the database is hosted) identifier (next 3 bytes), process id (next 2 bytes) and a random value (the last 3 bytes). The second 'id'is imported from the .csv file.

If we want to query the MongoDB to get a specific document, we have to specify criteria and conditions if necessary (e.g. a date range), that identify the document we want returned. If we want to get a specific field from a certain document, we also need to specify a projection. The queries are pretty flexible and allow us to modify results by imposing limits, skips, sorts etc.

The following query is used for querying by a date range, e.g. from 19.11. 2012 to 20.11. 2012:

```
> db.dataset_params.find({"date": ISODate("2012-11-19T00:00:00Z"),
        "date":ISODate("2012-11-20T00:00:00Z")})
```

The system responds with all the records in the database that are in the specified date range. This query also shows some unusal records where all the values are labelled as "NULL". This must have been an error while saving the data, so these records could pose a problem for the data analysis.

4.2 Python

Python is a clear and powerful object-oriented programming language. Most importantly, it is free and really easy to learn. The language is highly readable and has an uncluttered visual layout, similar to pseudo code The readability and simplicity are in part due to the fact that Python uses an indentation to delimit blocks of code, functions etc. Before you can do any Python programming, you need to install the Python interpreter on your computer. This is the program that reads Python programs and carries out instructions. To check if there is already a version of Python on the computer, type *python* into the terminal. If Python is already installed, the interpreter should respond with the version number on display. The version used in my work is Python 2.7.6. Any application is run by typing

```
python application_name.py
```

into the terminal or, just

```
application_name.py
```

if the application header contains the following text:

```
#!/usr/bin/env python
```

which specifies how to run the application.

4.3 PyMongo

PyMongo is the recommended package for working with the MongoDB from Python. The PyMongo is actually a Python MongoDB's driver. It can be installed with *pip*, which is a package manager used to install and manage Python packages.

```
pip install pymongo
```

PyMongo's basic job is to maintain a steady connection to the database while allowing all the work to be done in the Python programming language.

When working with PyMongo, we need to create a connection to the database; we do it with the following Python code:

```
import pymongo as mongo

collection = mongo.Connection('localhost', 27017).magicdb.dataset_params
```

Here we are establishing a connection to MongoDB database through the port 27017, where the database is locally hosted, to the collection named *dataset_ params* that is inside the *magicdb* database.

4.4 Bottle

Bottle is a fast, simple and lightweight micro web framework for Python [30], contained in a single Python file. A web framework is a collection of packages and/or libraries that makes the application writing process easier and promotes the reuse of existing code for common functions and classes. In order to better understand what a web framework is, we need to go over a few basic terms. Let us start with what happens when a user types an URL name to the browser's

address bar and presses Enter. The web page is transmitted to user's browser as HTML, a language that browsers are using to describe the content and structure of a web page. The web server is an application that is responsible for sending HTML to browsers. Also, the machine the web server is hosted on is also called a web server, so we have to be careful and try not to confuse the two.

Browser downloads a website from a web server using the HTTP protocol. The HTTP protocol is based on a request-response model. The client (a browser) requests data from a web application and the web application responds to the request with the data the browser has requested. The communication is always initiated by the client (the browser).

The messages in the HTTP protocol have methods associated with them. The HTTP methods correspond to different types of requests the client sends, which represent different intentions of the client. Requesting the HTML of a web page is different than submitting a form, so these two actions require the use of different methods. The GET method requests data from the web server, while the POST method allows us to send data to the application, e.g. via a form. The POST requests usually lead to a change in the application. For example, signing up for an account is done with the POST-ing of the form that user has filled with his desired account information.

All of this is true for the GET-ting of static HTML web pages, but it is also possible to create web pages dynamically such that they contain data retrieved from a database. The user generates a GET request from the web page by selecting date, telescope and telescope parameters that Bottle receives, processes and returns the data requested from the database, in the desired date range, plotted.

The Bottle framework was also installed with the pip package management system.

```
pip install bottle
```

The Bottle web framework is very simple and easy to learn but powerful enough to support running of a fairly sized web application. Because Bottle is an application, it shuts down when the user logs out, so the solution for this is making it run as a service. For this purpose, I wrote a *bash* script that is handling the automatic starting of the bottle service every time the server is rebooted. The shell script has to be located inside the /etc/init.d/ folder under the name *magicapi*. The command used for booting the bottle application as a service is:

```
sudo service magicapi start
```

4.5 Dygraphs

The Dygraphs is a javascript library used for graphs drawing and data visualisation. The library easily handles huge data sets and plots millions of points without showing signs of slowing down. Also, it is very interactive; users can zoom parts of the graph, both vertically and horizontally, and hover with the mouse over the lines to see the exact values. The graphs support error bars (confidence intervals) around data series, if there is a rms provided for each data point. The graphs are even zoomable on mobile devices!

Dygraphs accepts various document formats; array of arrays is the native format and is the one I chose to work with. An example of this 'array of arrays 'is: [[1,10,100], [2,20,80], [3,50,60], [4,70,80] etc.], { labels: ["rms", "mean", "median", etc.] }. Headers for this native format are specified through the labels option. When the date is on the x axis, the Date object is specified in the first column, like this: [[new Date('2010/07/12 '), 100, 200], [new Date('2009/07/19 '), 150, 220], etc.].

For the y- values, we can specify errorBars with the array format as well. This is formatted as follows: *errorBars*: [x, [value1, rms1], [value2, rms2], ...].

4.6 The application architecture

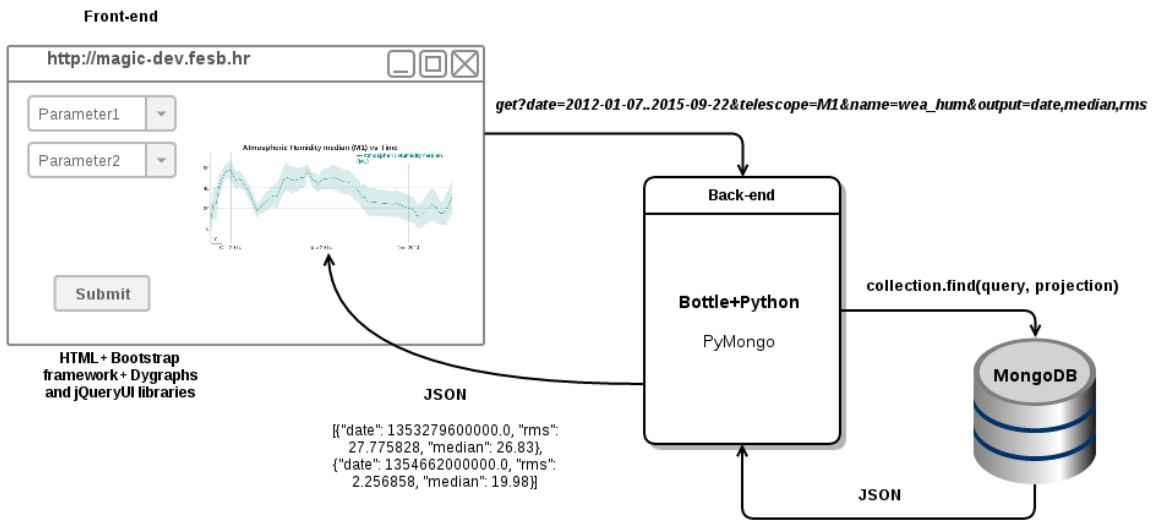

Figure 4.1: The application architecture. From: [29]

The web application is a single-page application (SPA), which means that after the first

page load, all subsequent page and content changes are handled internally by the application. That means that the browser never triggers a new page load. Interaction with the single page application involves a dynamic communication with the web server. The Figure 4.1 shows the architecture of the application; as any typical web application, it has three layers: the front-end, the back-end and the MongoDB database. In this case, the three layers are only logically divided and are all located on the same server. From the image we can see the approximate flow of the internal work of the application. When a user types the address $www.magic-dev.fesb.hr$ into the browser's address bar, the browser sends a GET request to the physical location of the web page for its content.

The front-end

The front-end part of the application is a layer of the application users interact with directly, from the browser. This includes the design of the web page as well as the logic behind the scenes, that communicates with the back-end. Tools used for front-end development are usually JavaScript, HTML and CSS. The objective of the site design is ensuring that when a user opens up the site, he sees the relevant information, in a format that is easy to read and/or is intuitive for use. The HTML (Hyper Text Markup Language) is the essential part of any website development process, that provides the structure of the site. The CSS (Cascading Style Sheets) controls the visual part of the site and allows customizing the design. JavaScript is an event-based programming language that is used to transform static HTML pages into a dynamic interface. JavaScript code can use the Document Object Model (DOM) to change the 'original'web page as a response to events. Bootstrap offers many of the popular UI components with an elegant style, a grid system and JavaScript plugins. It consists of several parts: global styles, responsive 12-column grids and layouts, basic CSS (fundamental HTML elements like tables, forms, buttons and images) styled and enhanced, collection of reusable components (drop downs, buttons, navigation controls etc.) and jQuery plug-ins.

The elements on the page that users communicate with are the drop down lists and the date picker. First, a user can select a parameter he wants to be plotted on the X-axis, the telescope which recorded the said parameter and, if possible, the *rms* check box. The process is repeated for the Y-axis of the graph. These are the mentioned elements:

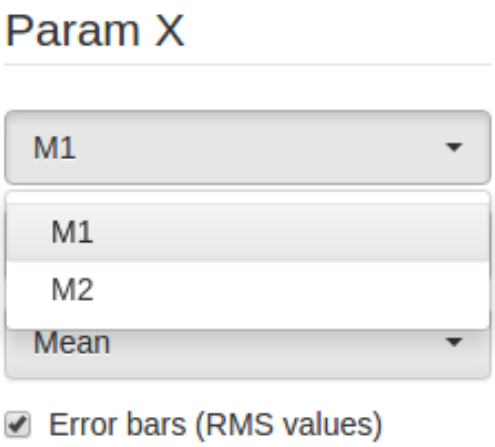

Figure 4.2: The telescope drop down list, where the user selects the desired telescope

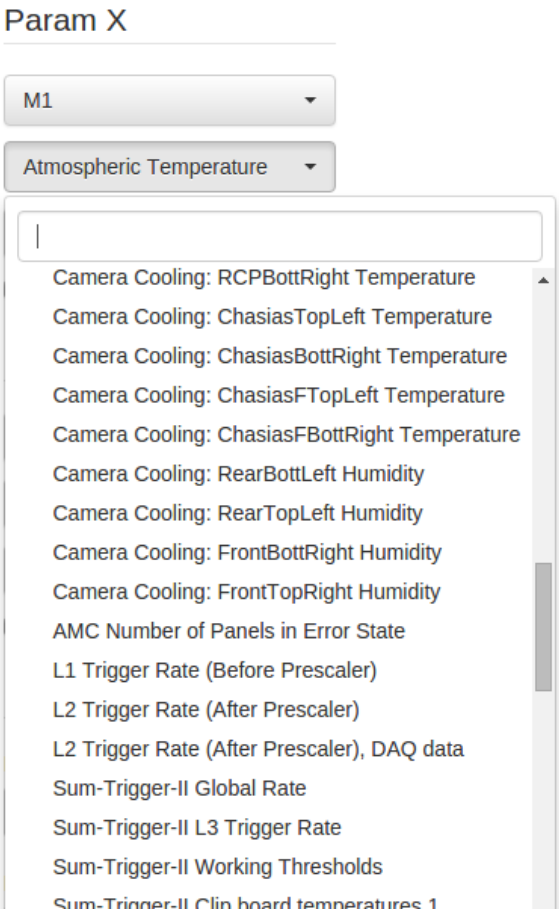

Figure 4.3: The parameter drop down list, contains all the parameters as well as a search bar to avoid too much scrolling

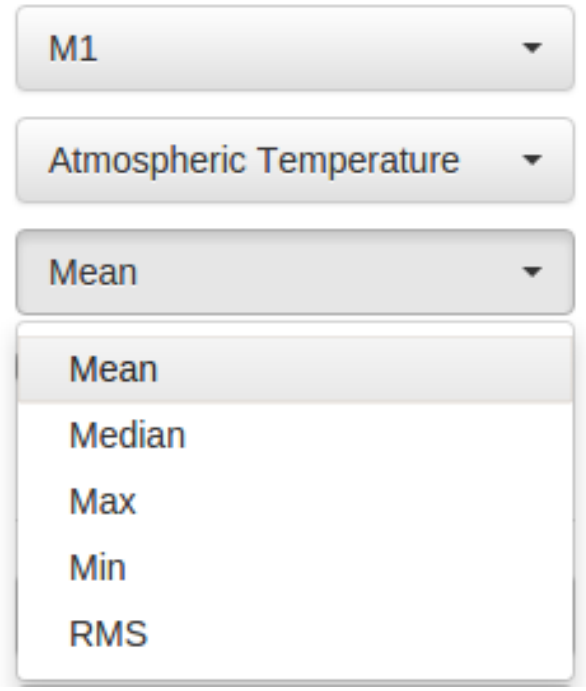

Figure 4.4: The descriptive statistics drop down list, contains min, max, median, rms, mean

Now the user has to select a date range for the desired parameters. Upon clicking on the date input field, an interactive calendar pops up, where the user can select day, month and year of interest. If the date is correctly chosen (the user has not clicked outside the calendar before picking a day), user can see the chosen date in the input field. This is the date picker:

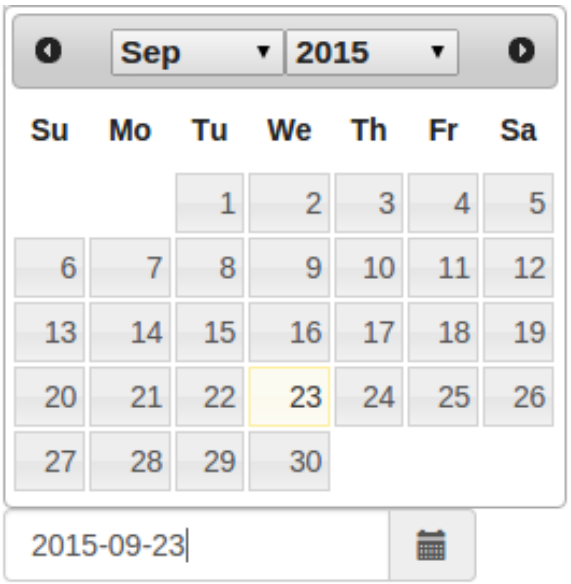

Figure 4.5: The date picker (From: [33])

When the user has selected all the parameters for the graph plotting, nothing happens unless the *Submit* button is pressed. This clicking on an HTML button is called an 'event' and it triggers an action or a code execution. After pressing the button, the JavaScript processes all

the entered parameters, generates a new query in the following format:

```
get?date=2012-01-07..2015-09-22&telescope=M1&name=wea_hum&output=date,median,rms
```

and forwards the request to the back-end part. The back-end responds, and the graphs appear on the screen.

The figure 4.6 is showing the whole front-end part of the application. They are zoomable vertically and horizontally and can be grouped by days. Also, the exact values of parameters (x, y) are shown upon mouse hover.

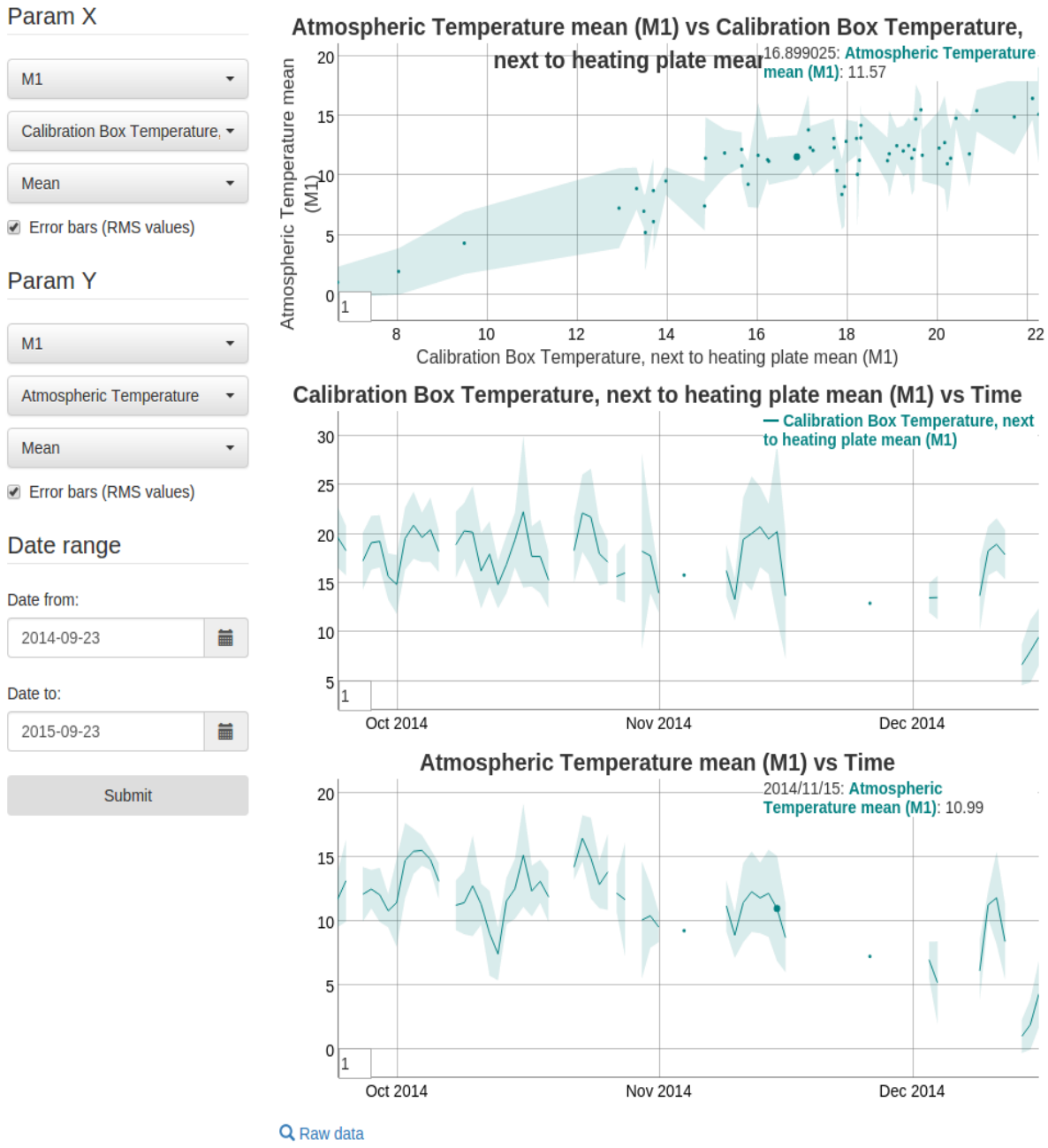

Figure 4.6: The front-end of the web application

Clicking on the small 'Raw data' link, user is redirected to a new browser window where he can copy all the data he entered into the application in order to get the desired graph.

Figure 4.7: The link that redirects to a new window where the user can copy all the data used for plotting the graph that is currently on screen

The back-end

The term 'back-end' refers to a piece of functionality that is a part of an application that is not directly accessed by the user. The backend usually consists of a server, an application and a database. This part of the application handles the database connection and communication. It is using the Bottle service for retrieving desired documents from the MongoDB using PyMongo and returning them in the JSON format.

Upon receiving a GET request from the front-end part, the server then responds with a POST request by sending back the HTML; the structure and appearance of the page. When the page is loaded, the user can start interacting with it. The request is now in the format that the back-end 'understands', but not the database. The PyMongo transforms it to the required format:

```
collection.find(query, projection)
```

and forwards it to the database. The database responds with the desired data in BSON format, which the back-end turns into JSON and forwards back to the JavaScript. The data returned in the JSON format look like this:

```
[{"date": 1353279600000.0, "rms": 27.775828, "median": 26.83},
{"date": 1354662000000.0, "rms": 2.256858, "median": 19.98},
{"date": 1354748400000.0, "rms": 5.639246, "median": 17.790001} ... ]
```

(BSON is a binary encoding for JSON. Like JSON, BSON supports the embedding of the documents and arrays within other documents or arrays. BSON also contains extensions that allow representation of data types that are not part of the JSON spec, for example BSON has a Date type.[31]) The data in JSON format needs to be formatted so that the dygraphs library can plot the graphs. The result is the plotting of three graphs, parameter X vs time, parameter Y vs. time and X vs Y.

Data Analysis

1 Introduction

All possible techniques for the data analysis are divided into **graphical** and **quantitative** ones. For the exploratory data analysis (EDA), the emphasis is on the graphical techniques: *scatter plots*, *character plots*, *box plots*, *histograms*, *probability plots*, *residual plots*, *mean plots* etc. The graphical tools are the shortest path to gaining insight into a data set in terms of testing assumptions, model selection, model validation, estimator selection, relationship identification, factor effect determination, outlier detection etc. The downside of the EDA is that these techniques are rather subjective and depend on the interpretation. This approach is trying to maximize insight into the main data characteristics, uncover underlying structure, detect outliers and anomalies, make models and build predictions. The EDA postpones assumptions about what kind of model the data follow, allowing the data itself to reveal the underlying structure.

Quantitative techniques include ANOVA, t-tests, chi-squared tests, F-tests etc., and they focus on statistical procedures that yield some numeric or tabular output. Examples of the techniques used are: hypothesis testing, analysis of variance, point estimates and confidence intervals, least squares regression etc. The numbers we get from doing a quantitative analysis provide us with some insight into the data. The mean and median are measures of central tendency that describe the most common value in a set of data. The *median* may be a better indicator of the most typical value if a set of data has outliers (extreme values that greatly differ from others). However, when the sample size is large and does not have outliers, the *mean* usually provides a better measure of central tendency. The summary values give us a measure of the amount of variability or spread in the data set. The most common measures of variability are the range, variance, and standard deviation. Sometimes, it is convenient to know the position of a certain value, relative to other values in a data set. The most common measures of position are quartiles and the standard score (or, the z-score).

In practice, analysts typically use a mixture of graphical and quantitative techniques. The usual steps that need to be followed in order to understand, clean and prepare the data for the building of a predictive model:

1. Variable identification
2. Univariate
3. Bivariate analysis

4. Treatment of missing values
5. Treatment of outliers
6. Variable transformation

Usually, one needs to iterate over steps 4 - 6 multiple times before a refined model is found.

2 Variable identification

The first step in any analysis is to identify the problem, or a question we are trying to answer with the data analysis. In my case, the wuestion I want to answer is: is the L1 trigger rate of the MAGIC telescopes in any way connected to the atmospheric conditions, such as cloudiness or temperature? My *Predictor* (input) and *Target* (output) variables would then be: the sky temperature an input variable and the trigger rate an output.

3 Univariate analysis

In this step, we are exploring our identified variables independently, one by one. To get a better understanding of a continuous variable, we need to understand its **central tendency** and **spread** of the data. We visualize the data using a *histogram*, *box plot* and *measures of dispersion* and *central tendency*. Measures of central tendency are: mean, median, mode, min and max. Measures of dispersion are range, quartiles, IQR, variance, standard deviation, skewness and kurtosis.

3.1 Measures of central tendency

Median is the middle value in a sorted data series; to find the median, we arrange data from the smallest to largest value. If there is an odd number of observations, median is the middle value. If there is an even number of data, median is the average of the two middle values.

Mean is an average, often denoted by $\bar{X}$. The mean of a set of N numbers is:

$$\bar{X} = \frac{\Sigma X_i}{N} \tag{3.1.1}$$

Mode is the value that appears most frequently in a population or a sample.

3.2 Measures of dispersion

Quartiles divide data into four equal parts. The values that divide each part are called the first (Q_1), second (Q_2), and third (Q_3) quartile. Q_2 is the median value in a data set.

Standard score

A standard score shows how many standard deviations an element is from the mean. We calculate it from the following formula.

$$z = (X - \mu)/\sigma \tag{3.2.2}$$

where z is the z-score, X is the value of the element, μ is the mean of the population, and σ is the standard deviation.

The range and interquartile range (IQR)

The range is a difference between the largest and smallest values in a set of data. It is a measure of where the first and last data items are in a set. The interquartile range is a measure of where the middle 50% od the data is in a data set. The IQR is used to measure how spread out the data points are from the mean of the data set. The higher the IQR, the more spread out the data points. Gives best results when considered with other measurements, the median or total range, to build a complete picture of the clustering tendency of the data around the mean.

The Variance

In a population, variance is the average squared deviation from the population mean. It is defined by the following formula:

$$\sigma^2 = \Sigma(X_i - \mu)^2/N \tag{3.3}$$

where σ^2 is the population variance, μ is the population mean, X_i is the i-th element from the population, and N is the number of elements in the population.

The Standard Deviation

The standard deviation is a square root of the variance. The formula for the standard deviation of a population is:

$$\sigma = \sqrt{\frac{\Sigma(X_i - \mu)^2}{N}} \tag{3.4}$$

where σ is the population standard deviation, μ is the population mean, X_i is the i-th element from the population, and N is the number of elements in the population.

Skew

The moment coefficient of skewness of a data set is:

$$skewness : g_1 = m_3/{m_2}^{3/2} \tag{3.5}$$

.

where $m_3 = \Sigma(x-\mu)^3/N)$ is the so-called *third moment* of the data set and $m_2 = \Sigma(x-\mu)^2/N)$ is the variance. If the resulting number is positive, the distribution is positively skewed (to the right), if it is negative then the distribution is negatively skewed (to the left). If it is equal to zero, then the data are perfectly symmetric. Negatively skewed distributions have a long left tail, while positive skew has a right tail.

A non-symmetrical or skewed distribution occurs when one side does not mirror the other. Non-symmetrical distributions are generally described as being either positively skewed or negatively skewed.

Kurtosis

Kurtosis refers to the degree of a peak in a distribution; is it sharp and high, or short and broad? This can be visually inspected from a histogram, but a numerical value is more precise. Higher value of kurtosis indicate a higher, sharper peak and lower value indicates a lower peak. The reference standard for kurtosis is a normal distribution, and it equals three. Moment coefficient of kurtosis of a data set is computed using the following formula:

$$\begin{aligned} kurtosis : a_4 &= m_4/{m_2}^2 \\ excess\, kurtosis : g_2 &= {a_4}^{-3} \end{aligned} \tag{3.6}$$

where $m_4 = \Sigma(x-\mu)^4/N$ and $m_2 = \Sigma(x-\mu)^2/N$.

3.3 Visualization methods

Histogram graphically summarizes the distribution of a univariate data set and is showing the following data properties: location of the center, spread of the data, presence of outliers and multiple modes. If the histogram indicates a symmetric, moderately tailed distribution, the next step is to do a normal probability plot to confirm the normality. The distribution can have one, two or more peaks. The one with only one clear peak is called unimodal (bell-shaped), and a distribution with two clear peaks is called bimodal. Some may have many more values on one side of the graph than the other; if there is fewer observations on the right - we say that the distribution is skewed right. Distributions with fewer observations on the left are said to be skewed left. When the observations in a set of data are equally spread across the range of the distribution, the distribution is called *uniform*.

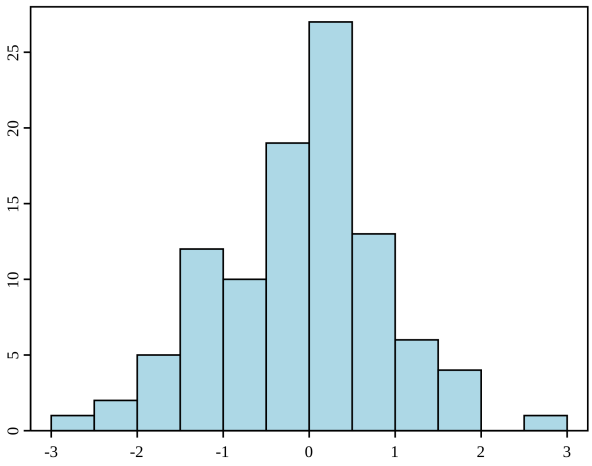

Figure 3.1: An example of a histogram (From: [34])

Box plots are a tool for checking whether there are location and variation shifts in a data set and provides insight into whether there are outliers in the data. Box plots are formed by putting the response variable on the vertical axis and the factor of interest on the horizontal axis, the data set split into quartiles. The body of the box plot consists of a 'box'which goes from the first (Q1) to the third quartile (Q3). Inside the box, a vertical line is drawn at the Q2, the median of the data set. Two other horizontal lines, *whiskers*, extend from the front and back of the box. The front whisker goes from Q1 to the smallest non-outlier in the data set, and the back whisker goes from Q3 to the largest non-outlier. If the data set includes outliers, they are plotted separately as points on the chart.

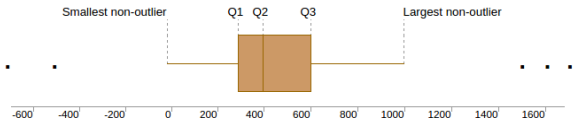

Figure 3.2: Boxplot explained (From: [36]

The box plot also provides information about the shape of the data set, some common patterns are:

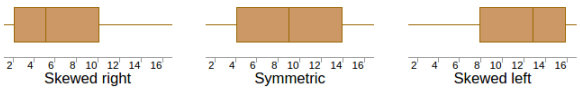

Figure 3.3: Common patterns in the shapes of data sets(From: [36]

Also, box plots display *range* and *interquartile range*. Interquartile range is represented by the width of the box (Q3 minus Q1).

4 Bivariate analysis

Bivariate analysis is the analysis of the relationship between two variables. A very tools for helping to visualize relationships between the data is a *scatter plot*. Each dot on the scatter plot represents one observation from a data set. We can describe patterns seen on a scatter plot in terms of linearity, slope, and strength. *Linearity* refers to whether the pattern is linear or non-linear, *slope* to the direction of change in variable Y when variable X increases (if Y also increases, the slope is positive; if it decreases, the slope is negative) and *strength* refers to the strength of the relationship; if the dots are widely spread in the plot, the relationship between variables is weak and if they are more concentrated, the relationship is strong. Scatter plots can also reveal unusual features in data sets, i.e. clusters, gaps or outliers. The plot does not indicate the strength of the relationship, to find it, we use *correlation coefficients*

Correlation coefficient describes the direction and the magnitude of the relationship of the two variables. The most common correlation coefficient is the Pearson product-moment correlation coefficient and it measures the strength of the linear association between variables. The sign and the absolute value of a correlation coefficient describe the direction and the magnitude of the relationship between two variables, so its value ranges between -1 and 1. The greater the absolute value of a correlation coefficient, the stronger the linear relationship. The strongest linear relationship is when the correlation coefficient is either -1 or 1, and the weakest when the correlation coefficient is equal to 0. A *positive correlation* means that if one variable increases, the other one increases as well and *negative correlation* means that if one variable increases, the other one decreases. The aforementioned Pearson product-moment correlation coefficient only measures linear relationships, so a correlation of 0 does not mean zero relationship between two variables, it means zero linear relationship. When the slope of the line in the plot is negative, the correlation is negative; and vice versa. The strongest correlations (r = 1.0 / r = -1.0) occur when data points fall exactly on a straight line. The correlation becomes weaker as the data points become more scattered. If the data points fall in a random pattern, the correlation is equal to zero. Correlation is greatly affected by outliers, so they have to be dealt with prior to the analysis. The most common correlation coefficient is the **Pearson product-moment correlation coefficient**, we calculate it using the following formula:

$$r = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{[\Sigma(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})^2)]}} \tag{4.1}$$

5 Missing values

Missing values in a data set occur when no data value is stored for an observation. There can be any number of reasons for this, technical problems during recording, extreme weather conditions, or simply human error. The values need to be dealt with because they can reduce the predicting power of a model derived from such data.

There are several methods used for dealing with missing values. We can simply **delete** them and exclude them from our data analysis, which is not always a good idea, because by doing so, one can loose a large proportion of important data. A better solution than deletion is to fill the values with estimated values. We use a known relationship between the rest of the data and use it to estimate the missing values. **Mean, mode and median imputation** are the most commonly used methods for replacing the missing data. The imputation can be *generalized*, in the sense that mean or median from the rest of the data is computed and imputed or it can be *similar* in the sense that we group data by some property (e.g. sex, age group), calculate individual averages and replace them based on the group association. We can use a **prediction model** to estimate the missing values. This can be done if we divide the data into two sets, a one with no missing values and other with missing values. The set with no missing values becomes a training set for the model while the data with missing values become the target variable. Using the model we populate the missing variables from the target set. Usually, regression, ANOVA, logistic regression etc. are used to do this kind of prediction.

6 Outliers

Outlier is term used to describe a value that extremely differs from other data, i.e. a number that is several orders of magnitude greater than others. When we encounter outliers, ideally, we should find out what is the reason for their appearance in the data, because the method of dealing with them depends on said reason. Causes can be natural and artificial. Artificial means that it is due to an error of some kind., e.g. a faulty equipment. Outliers are dangerous as they can drastically influence the data analysis results and produce useless models. Outliers typically increase variance, decrease normality etc. Most commonly used method of detecting outliers is visualization, the before mentioned methods are used: histograms, box plots and scatter plots. There are some numerical rules as well, e.g. everything that falls out of the range $[-1.5 * IQR, 1.5 * IQR]$ is an outlier, data that are more than three standard deviations from mean, etc. We deal with outliers in a similar way as we do with missing values. We either remove them or transform them.

7 Transformation

Transformation refers to the act of replacing a variable with a function, i.e. we replace a variable x with its square root. This transformations aims at changing the distribution of a variable or its relationship with other variables. Transformation is useful when we want to change the scale of a variable but not the distribution, transform complex non-linear relationships into linear ones. Transforming a data set to increase linearity is basically a trial-and-error process. Steps are as follows:

- Conduct an ordinary regression analysis on the raw data

- Construct a residual plot,
- If the plot pattern is random, do not transform data
- If the plot pattern is not random, continue
- Compute the coefficient of determination R^2
- Choose a transformation method (or methods)
- Transform the independent variable, dependent variable, or both, depending on the data set
- Conduct the regression analysis again, using the newly transformed variables
- Compute the coefficient of determination R^2, using the transformed variables
- If the transformed data's R^2 is greater than the raw data's R^2, the transformation was successful; if not, a different transformation method may give better results

The best transformation method depends on the nature of the original data. The only way to determine which method is best is to try every single one and compare the results (i.e. residual plots, correlation coefficients etc.). (From: [36])

When a residual plot reveals that a data set might be non-linear, it is possible to try to 'transform' the raw data to make it more linear, using some kind of mathematical operation to change its measurement scale. A **Linear transformation** preserves the linear relationships between variables and the correlation between x and y is not changed after the transformation (a linear transformation of variable x would be multiplying it or dividing by some constant number). A **Non-linear transformation** changes the linear relationship between variables (increases or decreases it) and, consequentially, changes the correlation between them (a non-linear transformation of the variable would be taking the square root of it).

In regression, a transformation to achieve linearity is a special kind of non-linear transformation. It is the kind of transformation that increases the linear relationship between two variables. There are many ways to achieve this, some common methods are summarized in the table below:

Method	**Transformations**	**Regression eq.**	**Predicted value**
Standard linear regression	None	$y = \beta_0 + \beta_1 x$	$\hat{y} = \beta_0 + \beta_1 x$
Exponential model	y = log(y)	$log(y) = \beta_0 + \beta_1 x$	$\hat{y} = 10^{\beta_0 + \beta_1 x}$
Quadratic model	y = sqrt(y)	$sqrt(y) = \beta_0 + \beta_1 x$	$\hat{y} = (\beta_0 + \beta_1 x)^2$
Reciprocal model	y = 1/y	$1/y = \beta_0 + \beta_1 x$	$\hat{y} = 1/(\beta_0 + \beta_1 x)$
Logarithmic model	x = log(x)	$y = \beta_0 + \beta_1 log(x)$	$\hat{y} = \beta_0 + \beta_1 log(x)$
Power model	y = log(y) x = log(x)	$log(y) = \beta_0 + \beta_1 log(x)$	$\hat{y} = 10^{\beta_0 + \beta_1 log(x)}$

Table 7.1: The methods of data transformation

Ideally, all these methods need to be tested on the data, to be sure that they do increase the linearity of the relationship. Testing the effect of a transformation method should involve checking residual plots and correlation coefficients.

8 Regression

Regression analysis is a form of predictive modelling technique that investigates the relationship between a dependent (target) and independent variable (predictor). As regression analysis estimates the relationship between two variables, there are multiple benefits of using it - indicates the significant relationships between dependent and independent variables, the strength of impact of the independent variable on a dependent one and allows comparing the effects of variables measured on different scales. All these help researchers evaluate the best set of variables to be used for building the best predictive model.

There are various kinds of regression techniques. The techniques are driven by the number of independent variables, type of dependent variables and shape of regression line. The techniques are:

1. Linear Regression
2. Logistic Regression
3. Polynomial Regression
4. Stepwise Regression
5. Ridge Regression
6. Lasso Regression
7. Elastic Net Regression
8. Robust Regression

Important thing to note is that linear regression doesn't show causation, regression shows if the data are correlated, and if they are, by how much!

8.1 Linear Regression

Linear Regression, also known as Ordinary Least-Squares (OLS) Regression, is one of the most powerful and most commonly used techniques in statistical data analysis, applied to determining whether there is a relationship between dependent and independent variables. It is also one of the oldest, dating back to the eighteenth century. With this data analysis technique, in which a variable or variables, are used to predict the behaviour of others, we hope to find that a certain variable (e.g. Y) is varying as a straight-line function of another variable (e.g. X). Simple linear regression is appropriate when the following conditions are satisfied:

- The dependent variable Y has a linear relationship to the independent variable X. To check this, we have to make sure that the XY scatter plot is linear and that the residual plot shows

a random pattern,

- For each value of X, the probability distribution of Y has the same standard deviation σ. When this condition is satisfied, the variability of the residuals will be relatively constant across all values of X, which is easily checked in a residual plot,

- The Y values must be independent for any given value of X and are roughly normally distributed (i.e. symmetric and unimodal). A little skewness is ok if the sample size is large. A histogram will show the shape of the distribution.

The ordinary linear regression assumes that the relationship between two predictor (independent) and response (dependent) variables takes the form of a straight line:

$$Y = \beta_0 + \beta_1 X_1 + \epsilon. \tag{8.1}$$

(a line that best represents the data), where the Y is the dependent variable, X_1 is the independent variables, the coefficient β_1 slope and the constant β_0 an intercept. ϵ is an error term, that describes the amount of variation not predicted by the slope and intercept terms.

Obviously, it is not possible to find a line that will go through all of the points but the line should minimize the sum of squared vertical deviation from each point to the line. This vertical distance between each observation and the line that fits the regression line is called *error*. Using the OLS we calculate values of our parameters by minimizing the sum of the squared errors for all observations. The slope β_0 indicates the unit changes in y for every unit change in x.

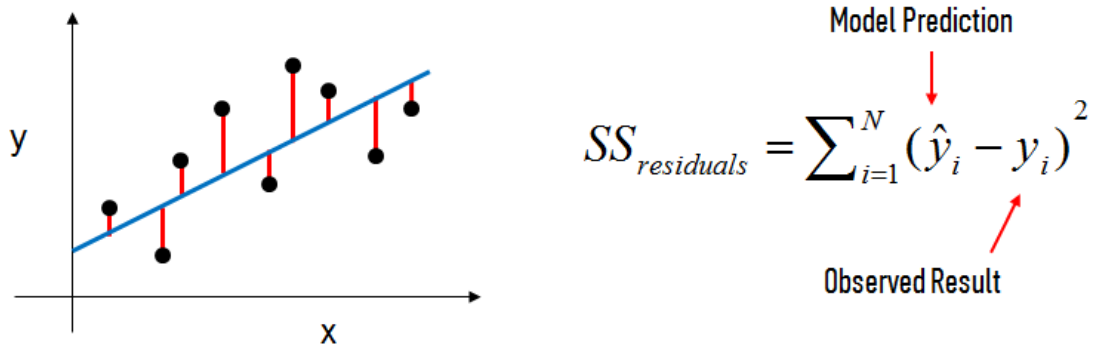

Figure 8.1: The least squares method - the black dots are **observed** values of x and y, the blue line is the **least squares line** and the red lines are the **residuals** (From: [38])

The model coefficients relate to the regression line:

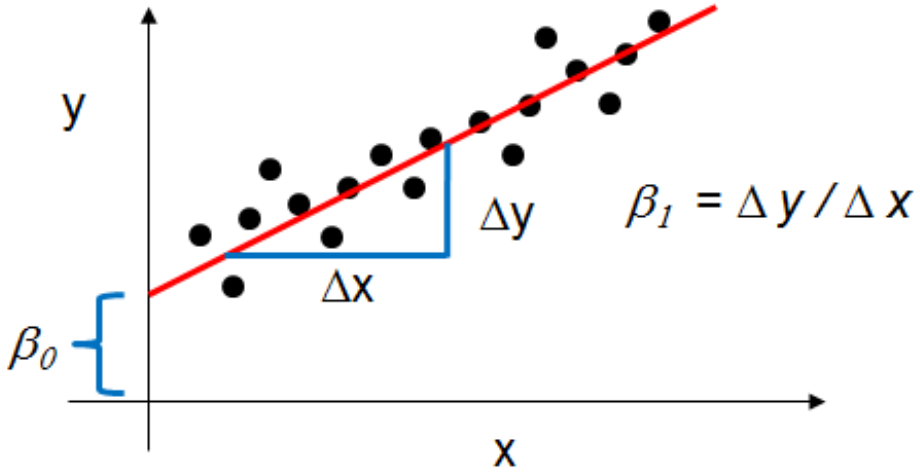

Figure 8.2: The least squares line relation to the coefficients (slope and intercept) (From: [38])

Testing the linear model:

Coefficient of determination

This statistic is the key output of the regression analysis; it gives us an indication of how well our model serves as an estimator of values for the dependent variable. We compute it by squaring the correlation coefficient. Since the correlation coefficient is given by r, the coefficient of determination is known as R^2. The values of R^2 range between 0 and 1; when it is 0, that means the dependent variable can not be predicted from the independent one, when it is 1, that means the dependent variable is perfectly predicted from the independent one and when it is somewhere in between, that indicates the extent of the predictability of the dependent variable.

The formula for computing the coefficient of determination for a linear regression model with one independent variable is given below:

$$R^2 = [(1/N)\Sigma[(x_i - \bar{x})(y_i - \bar{y})]/(\sigma_x \sigma_y)]^2 \tag{8.2}$$

where N is the number of observations used to fit the model, Σ is the summation symbol, x_i is the x value for observation i, $\bar{x}$is the mean x value, y_i is the y value for observation i, $\bar{y}$ is the value of y mean, σ_x is the standard deviation of x, and σ_y is the standard deviation of y.

Standard error of estimate

The standard error of the regression line is a measure that gives an indication of how well our linear regression model is working. It compares actual values in the dependent variable Y to the predicted ones. It gives us an indication of the predictive quality of a regression model;

a lower error indicates that more accurate predictions are possible. However, it does not indicate the extent to which the independent variable explains variations in the dependent model.

Residuals

The difference between an observed value of the dependent variable (y) and the predicted value ($\hat{y}$) is called the **residual** ($e = y - \hat{y}$). Each data point has a residual. Both the sum and the mean of the residuals are equal to zero; Σ e $= 0$ and $\bar{e} = 0$. A residuals plot is showing residuals on the Y - axis and the independent variable on the X - axis. If the points on a residual plot are randomly distributed around the horizontal axis, that is an indication that the linear regression model is appropriate for the data; otherwise, a non-linear model could be more appropriate.

8.2 Robust regression

Robust regression is a type of regression analysis that is searching for a relationship between dependent and independent variables in a data set, just like an ordinary regression, but where data is contaminated with outliers.

9 Making predictions

A very important part of the data analysis procedure is making predictions from the available data. A new discipline called **Machine Learning** is dealing with the studying of the methods of pattern recognition in data sets and creating algorithms that learn from the data, in order to be able to make predictions. There are many methods of machine learning, and are related to the nature of the data and predictive model one is trying to build. Depending on the type of the data and the model, the learning problems can be divided into *Supervised learning* and *Unsupervised learning*. Supervised learning methods are methods where training sets contain values targeted for prediction. From those values, the model is instructed how to make predictions and then compare them to the actual values in order to estimate if the model fits. Unsupervised learning methods are methods in which training sets consist of series of inputs without any target values and are clustering them in order to find groups of similar data in a data set.

The methods of robust regression we are going to use are RANSAC and Theil-Sen estimation. Random sample consensus (RANSAC) is a method of estimating model parameters from a data sample that contains outliers. The assumption on which it is based is that the data in the sample contain 'inliers', the data that can be explained by a set of model parameters, and 'outliers', the data that do n0t fit to the model. The algorithm repeats two steps during the fitting; first step is selecting a random sample of the data from the provided data set, from which the model parameters are computed and second, checking which elements in the provided data set are in agreement with the model estimated in the first step. The data

that do not fit into the model are considered outliers. Theil-Sen is a popular technique used for computing robust estimates of a linear regression coefficients between two variables in a data set.

10 The results of the data analysis

10.1 Python packages used for the data analysis

There is an abundance of open source libraries present that can be used for data analysis and machine learning for the Python programming language. Some of them are NumPy, SciPy, Pandas, Statsmodels, scikit-learn etc. **Matplotlib** is a library used for plotting a variety of graphs; histograms, line plots, scatter plots, box plots, heat plots etc. A nice feature of this library is that it allows the use of Latex commands for adding math to plots. **NumPy** stands for Numerical Python and has a powerful feature, the n-dimensional array. This library also contains basic linear algebra functions, Fourier transforms, advanced random number capabilities and tools. **SciPy** stands for Scientific Python and it was built on NumPy. This library is useful for a variety of high level science and engineering modules. SciPy and NumPy are great tools to provide most functionalities needed for extensive data analysis. However, tools with more advanced options were needed which led to the abundance of tools we have today. Examples of more advanced libraries are **Scikit Learn** and **Statsmodels**. Scikit Learn is a library used mostly for machine learning and was built on NumPy, SciPy and Matplotlib Contains a lot of efficient tools for machine learning and statistical modelling and is relatively easy to use and enables non-specialists to use machine learning algorithms. I used if tor robust linear regression, as it is the only library that enables the use of RANSAC and Theil-Sen estimators. **Statsmodels** is a module for statistical modelling, allows users to explore data, estimate statistical models, overall it is a module that allows users to perform the most extensive statistical testing. **Pandas** is a module used mostly for structured data operations and manipulations, that is for data munging and preparation. I used Pandas in order to read the .csv file into its DataFrame format, which has the shape (n, p) where n is the number of data points and p is the number of columns. Our y is a pandas series of length n. The Pandas module is very useful for the necessary data manipulations, i.e. excluding missing data or zero values.

10.2 Variable identification

The parameters of interest for the analysis are the Sky Temperature, recorded with a pyrometer device attached to the MAGIC telescope, and the L1 trigger rate of the telescope. The data is downloaded from the web application by selecting a date range from 19.12. 2012 - 15.9. 2015. and clicking on the link 'Raw data'. The data are then copy/pasted into a text editor where all

the spaces are replaced with commas to be saved in the .csv format. The .csv data file contains 530 measurements of the sky temperature and L1 trigger rate. Some of the measurements contain '0' values.

The independent variable (predictor) is the sky temperature and the dependent (target) is the L1 trigger rate.

10.3 The univariate analysis

Let us visualize both of our variables, separately, using a box plot and a histogram. The Python code used to plot the histograms:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#Reading the dataset into a dataframe using Pandas
df = pd.read_csv('l1t-mean-M1-pyro_skyt-mean-M1-p1etrue-p2etrue.csv',
        index_col=0, usecols = ["date","l1t_mean","pyro_skyt"])
X = df['pyro_skyt']
y = df['l1t_mean']

fig = plt.figure()

#Create one or more subplots using add_subplot
ax = fig.add_subplot(121)

#Labels and Title
plt.title('L1 Trigger Rate')
plt.xlabel('L1 Trigger Rate')
plt.ylabel('Count')
ax.hist(y, bins = 10, range = (y.min(),y.max()))

ax = fig.add_subplot(122)
plt.title('Sky Temperature')
plt.xlabel('Sky Temperature')
plt.ylabel('Count')
ax.hist(df['pyro_skyt'], bins = 10, range = (X.min(),X.max()))

plt.show()
# Save the figure...
fig.savefig('histograms_before_treatment.png')
```

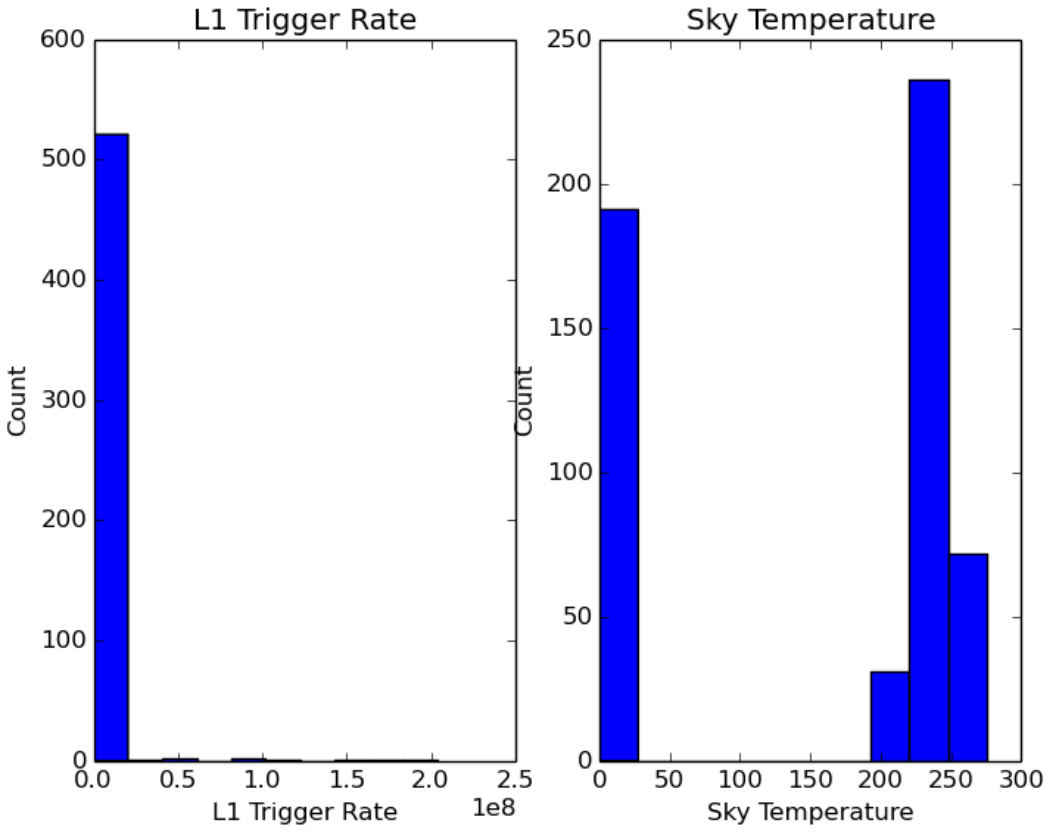

Figure 10.1: The histograms

We can see that there is a huge amount of zero values in the data set. This will surely affect the box plots greatly.

```
import numpy as np
import matplotlib.pyplot as plt
import pylab
import pandas as pd

#Reading the dataset into a dataframe using Pandas
df = pd.read_csv('l1t-mean-M1-pyro_skyt-mean-M1-p1etrue-p2etrue.csv',
        index_col=0, usecols = ["date","l1t_mean","pyro_skyt"])

fig = plt.figure()

ax = fig.add_subplot(121)
plt.title('L1 Trigger Rate')

df.sort()
df.boxplot(column='l1t_mean')

ax = fig.add_subplot(122)
```

```
19 plt.title('Sky Temperature')
20 df.sort()
21 df.boxplot(column='pyro_skyt')
22 plt.show()
23 # Save the figure...
24 fig.savefig('boxplots_before_treatment.png')
```

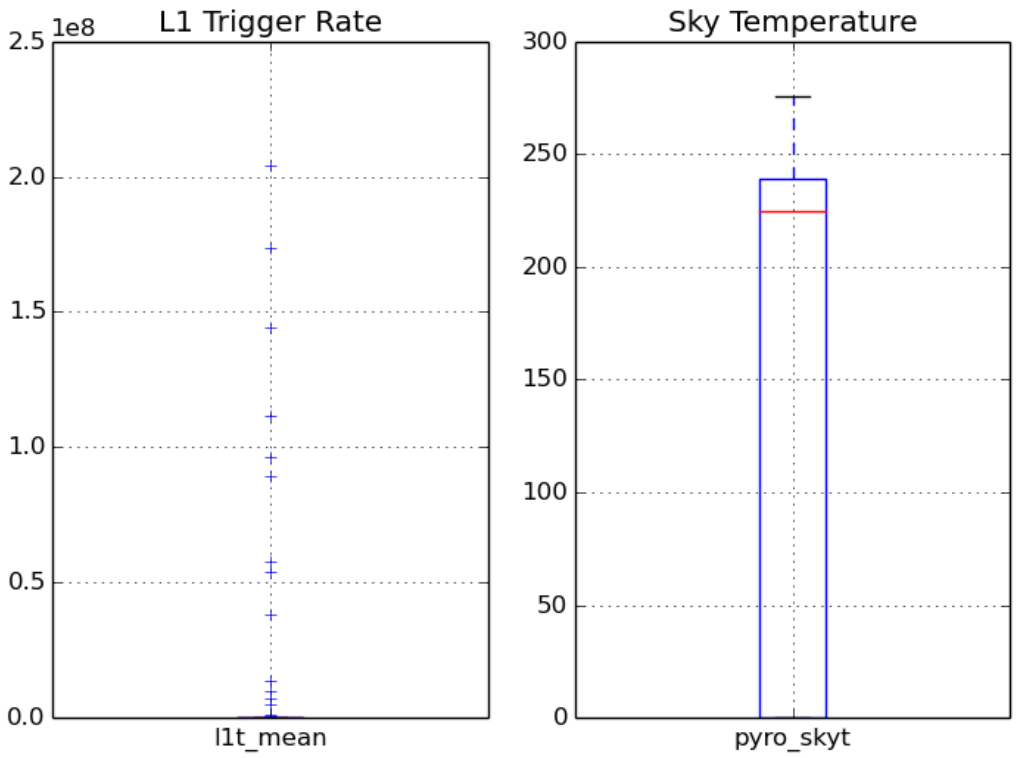

Figure 10.2: The box plots

We can see that the trigger rate values have such huge outliers, that we can not even see the box. As for the sky temperature, the only thing we can se, is that the data is not symmetrical. However, there are numerous zero values in the data set, so we are going to exclude them and plot again. This can be done by inserting the following command in the previous script:

```
1 data = data[(data == 0).any(axis=1)]
```

Now our histogram and box plot look like:

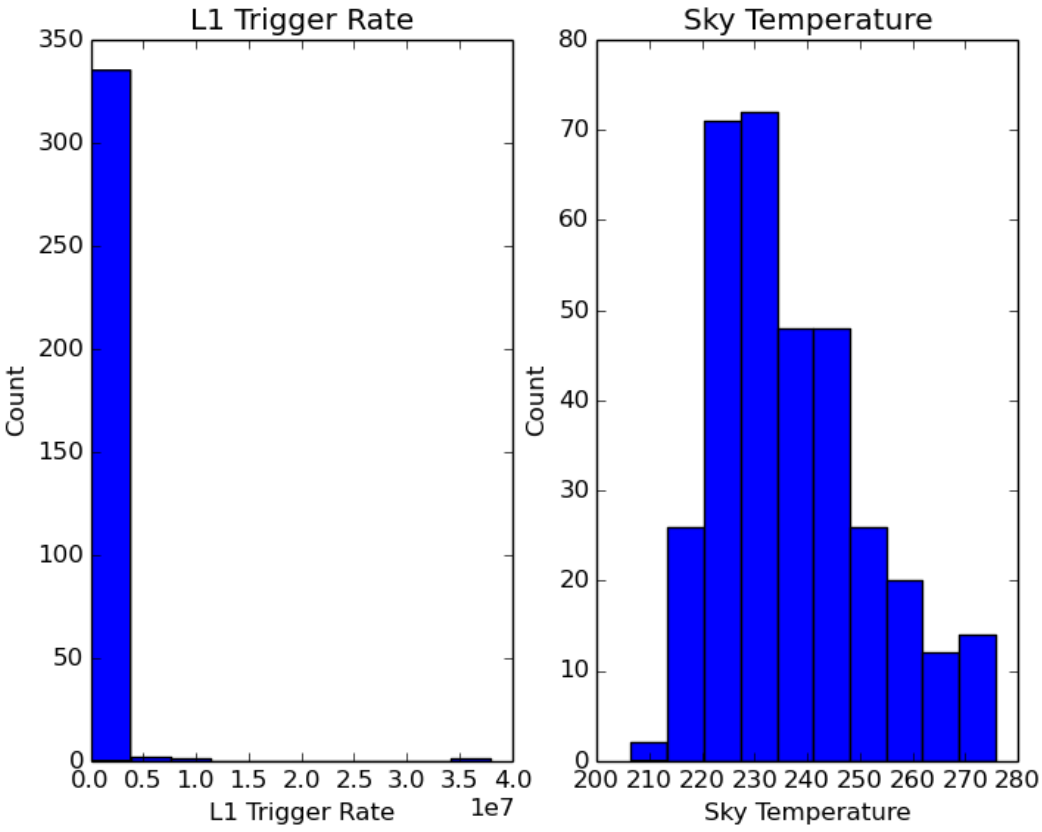

Figure 10.3: The histograms with excluded zero values

The trigger rate's histogram is not really giving any insight into the data, but the sky temperature's histogram is at least showing possible normality and a skewness to the right.

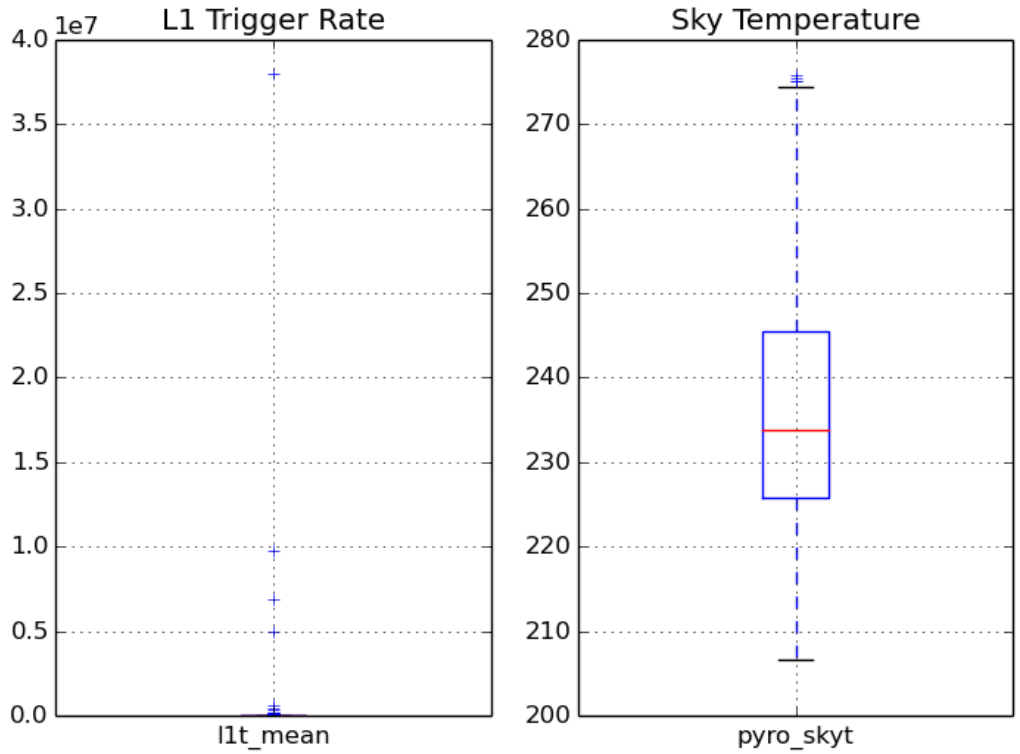

Figure 10.4: The box plots with excluded zero values

The box plot for the trigger rate is still not visible, but for sky temperature we have stronger sense of the outliers, it is central tendency etc.

We should see some numbers describing the data. Python code for getting the descriptive statistics:

```
1  import pandas as pd
2
3  #Reading the dataset into a dataframe using Pandas
4  df = pd.read_csv('l1t-mean-M1-pyro_skyt-mean-M1-p1etrue-p2etrue.csv',
5          index_col=0, usecols = ["date","l1t_mean","pyro_skyt"])
6
7  #Prints out  summary statistics for numerical fields
8  print('Describe----------------------------------------------')
9  print df.describe(), df.median(), df.kurtosis(), df.skew(), df.var()
```

The output is shown in the terminal:

	Sky Temperature	L1 Trigger Rate
mean	237.210966	201049.441362
std	14.606702	2172295.403445
min	206.564920	0.004789
25%	225.879809	7338.965680
50%	233.749118	13721.990360
75%	245.451990	25040.432182
max	275.733222	37939917.443643
median	233.749118	13721.990360
kurtosis	-0.161238	272.787269
skew	0.726335	15.979150
variance	2.133557e+02	4.718867e+12

Table 10.2: The descriptive statistics for the data

The trigger rate with its huge outliers is very problematic, so it is definitely a variable that is going to have to be transformed for later analysis. But first, we are going to try to visualize the relationship between the two variables.

10.4 The bivariate analysis

The Python code for making a scatter plot:

```
#import pandas for data manipulation
import pandas as pd

#import matplotlib for visualisation
import matplotlib.pylab as plt

#load data
data = pd.read_csv('l1t-mean-M1-pyro_skyt-mean-M1-p1etrue-p2etrue.csv',
        index_col=0, usecols = ["date","l1t_mean","pyro_skyt"])
df = df[~(df == 0).any(axis=1)]

#assign columns to variables
X= data['pyro_skyt']
y = data['l1t_mean']

#Visualize
fig = plt.figure()
ax = fig.add_subplot(111)

#Labels and Title
plt.title('Data visualization')
```

```
22  plt.ylabel('L1 Trigger Rate')
23  plt.xlabel("Sky Temperature")
24
25  ax.scatter(X,y)
26
27  # Save the figure...
28  fig.savefig('scatter_no_zeroes.png')
```

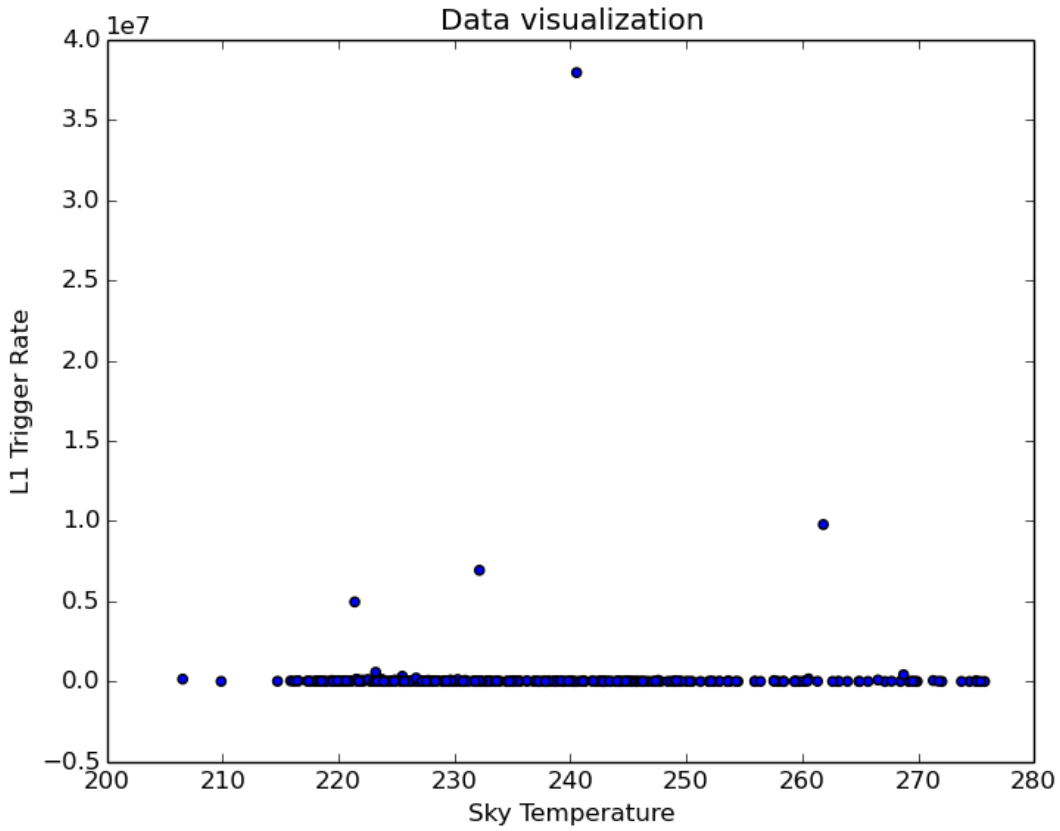

Figure 10.5: The scatter plot with excluded zero values

We can see a couple of extreme values of the trigger rate that are problematic for the visualization. I am going to try imposing a limit on the plot:

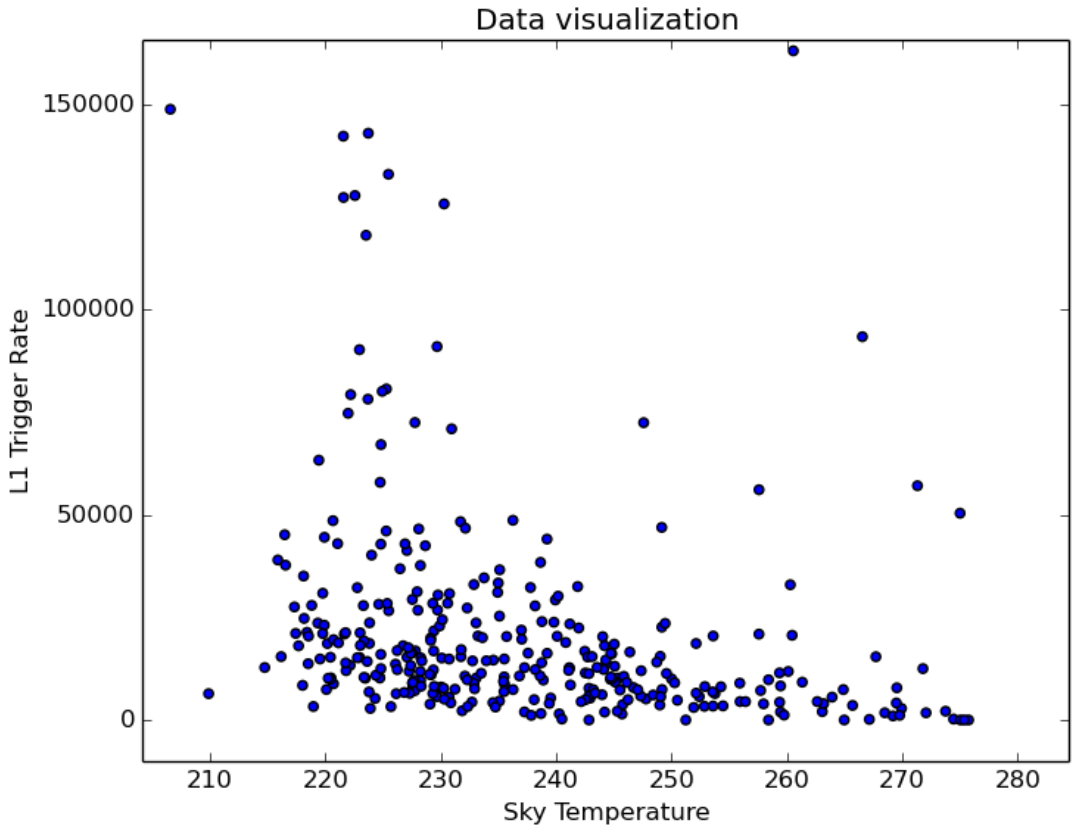

Figure 10.6: The scatter plot, with a limited y-axis

Now we can certainly see the data more clearly, but still can not make a conclusion as to what sort of relationship this might be. As mentioned before, the trigger rate variable needs transformation.

10.5 Transforming the variables

I tried all of the transformations from the Table 7.1 for the trigger rate, and none of them showed as much promise as $y = ln(y)$.

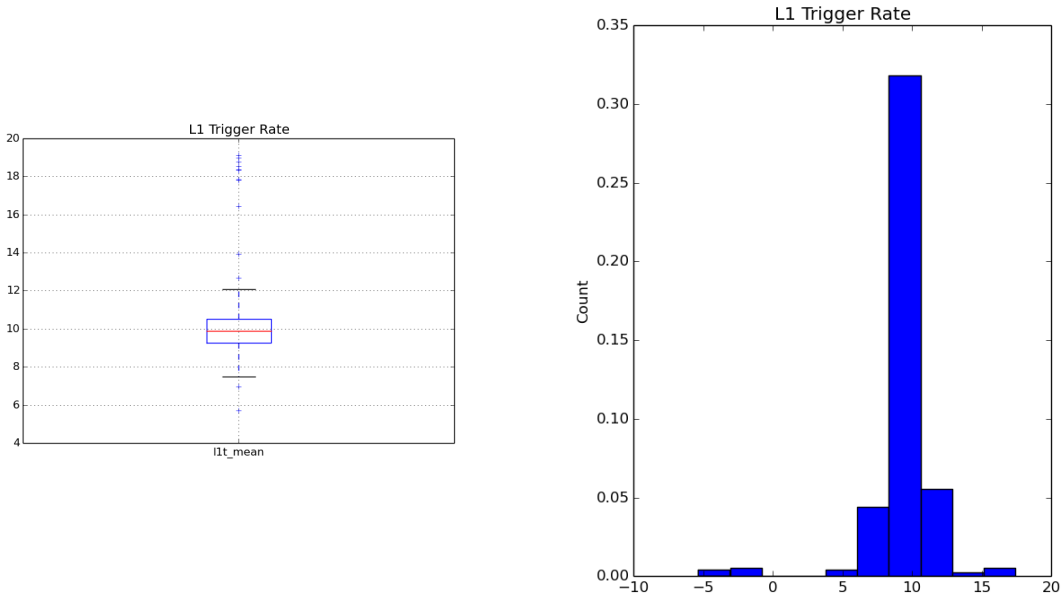

Figure 10.7: The box plot and histogram for the transformed trigger rate

The histogram looks like it is normally distributed, with a small tail on the left side. The box plot of the transformed trigger rate gives us a clearer picture about the median value, data spread and outliers.

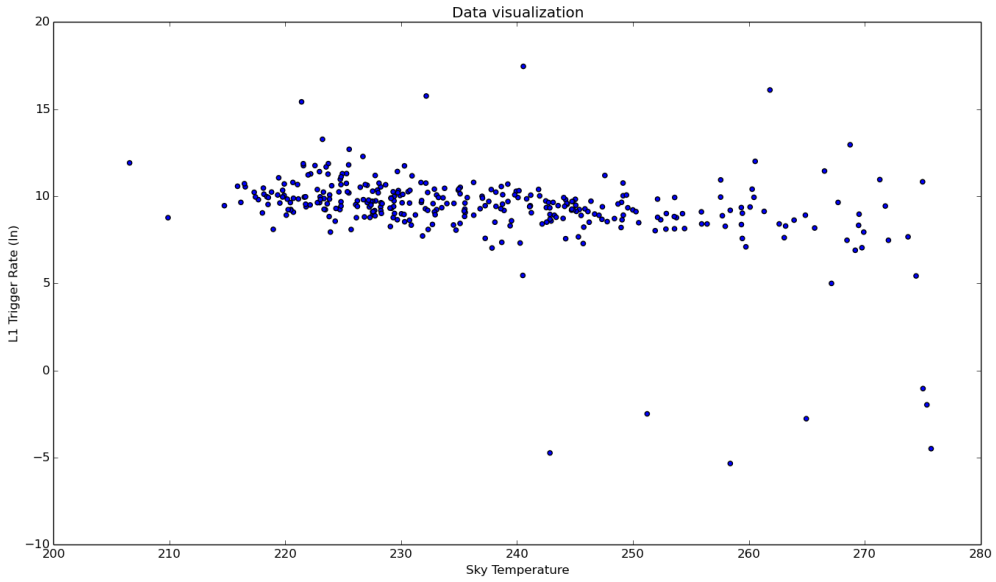

Figure 10.8: The scatter plot for y=ln(y) transformation

The scatter plot is now showing a possible linear relationship between most of the data. There are still a number of outliers that could pose a problem for the linear regression, but luckily, there are some very good tools for robust regression, that are taking care of the outliers.

We will see what the results of the ordinary linear regression will be.

10.6 Linear regression

The fitting of the regression line can be done in all of the aforementioned Python libraries for data analysis. The following code is for the simplest one, NumPy's *polyfit*: (an excellent choice for analysis is the *statsmodels OLS fit* as well; its output contains a great number of descriptive statistics)

```
import numpy as np
import matplotlib.pyplot as plt
import pylab
import pandas as pd

#Reading the dataset into a dataframe using Pandas
df = pd.read_csv('l1t-mean-M1-pyro_skyt-mean-M1-p1etrue-p2etrue.csv',
        index_col=0, usecols = ["date","l1t_mean","pyro_skyt"])

#Excluding rows that have in either column a value 0
df[~(df == 0).any(axis=1)]

X = df['pyro_skyt']
y = df['l1t_mean']

#Fit with np.polyfit
m, b = np.polyfit(X, y, 1)

#Visualize
fig = plt.figure()

plt.title('L1 Trigger Rate vs. Pyrometer: Sky Temeprature')
plt.ylabel('L1 Trigger Rate')
plt.xlabel('Sky Temperature')
plt.plot(X, y, '.')
plt.plot(X, m*X + b, '-')

plt.plot(X, y, '.')
plt.plot(X, m*X + b, '-')
#plt.show()
fig.savefig('fit.png')
```

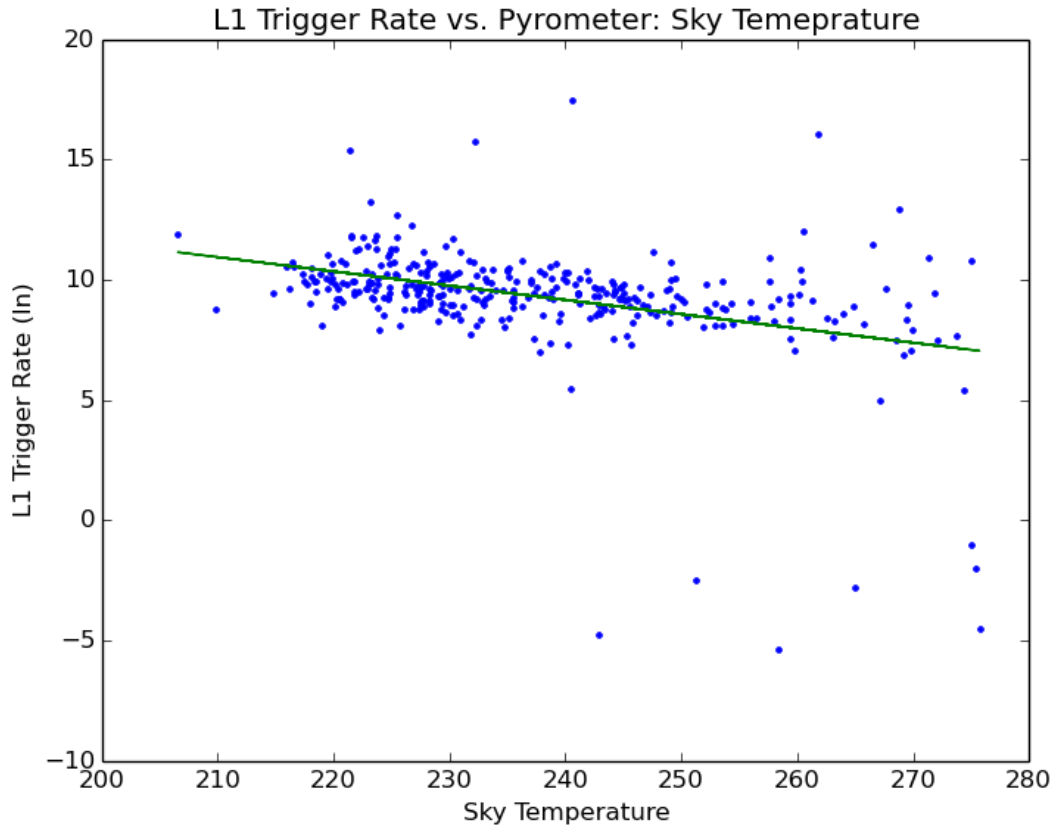

Figure 10.9: The fitted regression line

From the scatter plot, we can see that the regression line seems to be really good for most of the data.

The summary of the linear regression :

Dep. Variable:	y	**R-squared:**	0.147
Model:	OLS	**Adj. R-squared:**	0.144
Method:	Least Squares	**F-statistic:**	58.00
Date:	Sun, 20 Sep 2015	**Prob (F-statistic):**	2.67e-13
Time:	17:06:50	**Log-Likelihood:**	-730.16

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
Intercept	23.4037	1.851	12.644	0.000	19.763 27.045
X	-0.0593	0.008	-7.616	0.000	-0.075 -0.044

Table 10.3: Coefficients

Omnibus:	246.197	**Durbin-Watson:**	1.599
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5239.181
Skew:	-2.717	**Prob(JB):**	0.00
Kurtosis:	21.477	**Cond. No.**	3.87e+03

Table 10.4: Residuals

The results from the regression indicate that our intercept and slope coefficients are: $Intercept = 23.403749$, $slope = -0.059314$. That means that the regression line is described by the following formula:

$$y = (23.404 \pm 1.851) - (0.590 \pm 0.008)x \tag{10.1}$$

The right part of the table is showing the goodness of the fit. $R - squared$ is the coefficient of determination, a measure of how well the regression line approximates the real data points. F-statistic is a measure of how significant the fit is. t-statistic is a measure of how significant the coefficient is. Based on these values, the linear regression line seems like a really bad fit for this relationship. It is possible that we are missing one or more variables to get better results, or that the outliers are affecting the fit too much. One more thing to try is robust regression, the one that is resistant to outliers and see what we will get.

Robust regression

First let us take our data set, divide it into *train* and *test* sets and do the ordinary least regression. The Python code:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
import pandas as pd

# Load the dataset
data = pd.read_csv('l1t-mean-M1-pyro_skyt-mean-M1-p1etrue-p2etrue.csv',
        index_col=0, usecols = ["date","l1t_mean","pyro_skyt"])
data = data[~(data == 0).any(axis=1)]

magic_X= data['pyro_skyt'][:,np.newaxis]
magic_y = data['l1t_mean']
magic_y = np.log(magic_y)

# Split the data into training/testing sets
magic_X_train = magic_X[:-100]
magic_X_test = magic_X[-100:]

# Split the targets into training/testing sets
```

```
magic_y_train = magic_y[:-100]
magic_y_test = magic_y[-100:]

# Create linear regression object
regr = linear_model.LinearRegression(fit_intercept = True)

# Train the model using the training sets
regr.fit(magic_X_train, magic_y_train)

# The coefficients
print('Coefficients: %.2f' % regr.coef_)
print('Intercept: %.2f' % regr.intercept_)
# The mean square error
print("Residual sum of squares: %.2f"
      % np.mean((regr.predict(magic_X_test) - magic_y_test) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr.score(magic_X_test, magic_y_test))

# Plot outputs
fig = plt.figure()
plt.title('Pyrometer Sky Temperature vs. L1 Trigger Rate')
plt.scatter(magic_X_test, magic_y_test,  color='black')
plt.plot(magic_X_test, regr.predict(magic_X_test), color='blue',
         linewidth=1)
plt.show()
fig.savefig('ols.png')
```

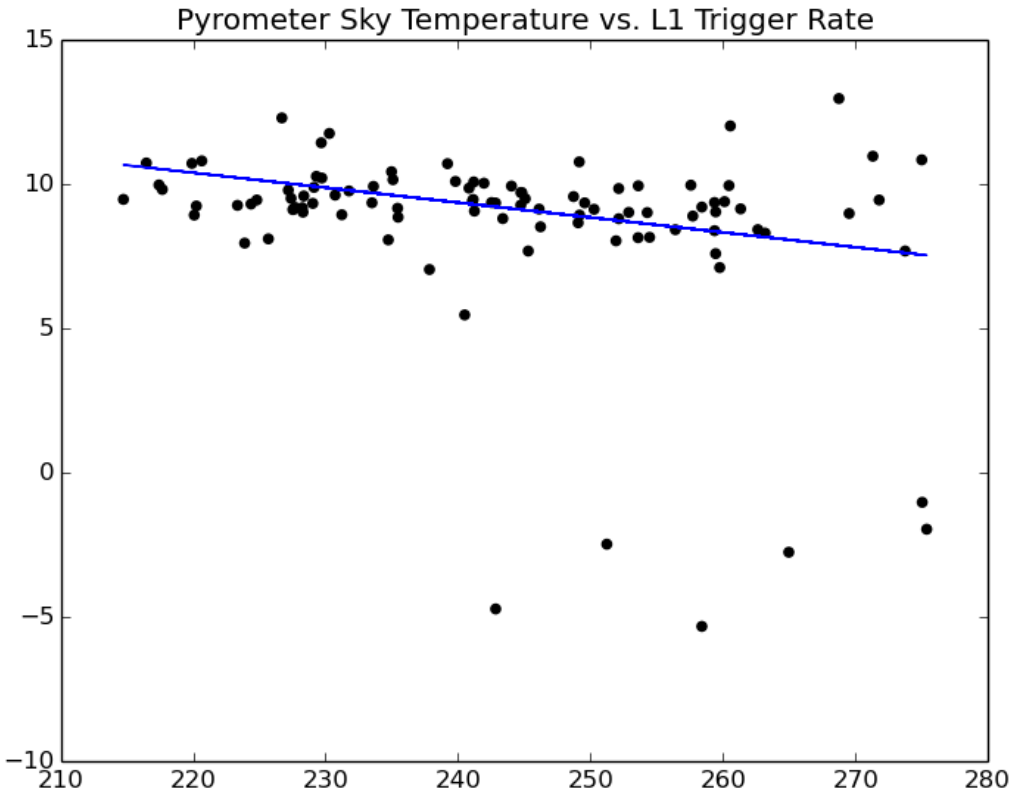

Figure 10.10: The repeated OLS by splitting data to training/test sets

Results of the OLS:
Coefficients: -0.05
Intercept: 21.75
Residual sum of squares: 9.42
Variance score: 0.06

We can see that these results are not that different from previous calculations. Now we shall use the robust regressors. As previously mentioned, RANSAC and Theil-Sen estimators were used. The Python code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, TheilSenRegressor,
        RANSACRegressor
import pandas as pd

estimators = [('OLS', LinearRegression()),
              ('Theil-Sen', TheilSenRegressor()),
              ('RANSAC', RANSACRegressor())]

#Reading the dataset into a dataframe using Pandas
```

```
df = pd.read_csv('l1t-mean-M1-pyro_skyt-mean-M1-p1etrue-p2etrue.csv',
        index_col=0, usecols = ["date","l1t_mean","pyro_skyt"])
#excluding rows that have in either column a value  0
df = df[~(df == 0).any(axis=1)]

#Assign columns to variables
X = df['pyro_skyt'][:,np.newaxis]
y = df['l1t_mean']
y = np.log(y)

linreg = LinearRegression(fit_intercept = True)

fig = plt.figure()
plt.title('Pyrometer Sky Temperature vs. L1 Trigger Rate (Before Prescaler)')

for name, estimator in estimators:
    estimator.fit(X, y)
    y_pred = estimator.predict(X)

    plt.plot(X, y_pred,
             label='%s' % (name))

plt.axis('tight')
plt.legend(loc='upper left')
plt.show()
fig.savefig('allest.png')
```

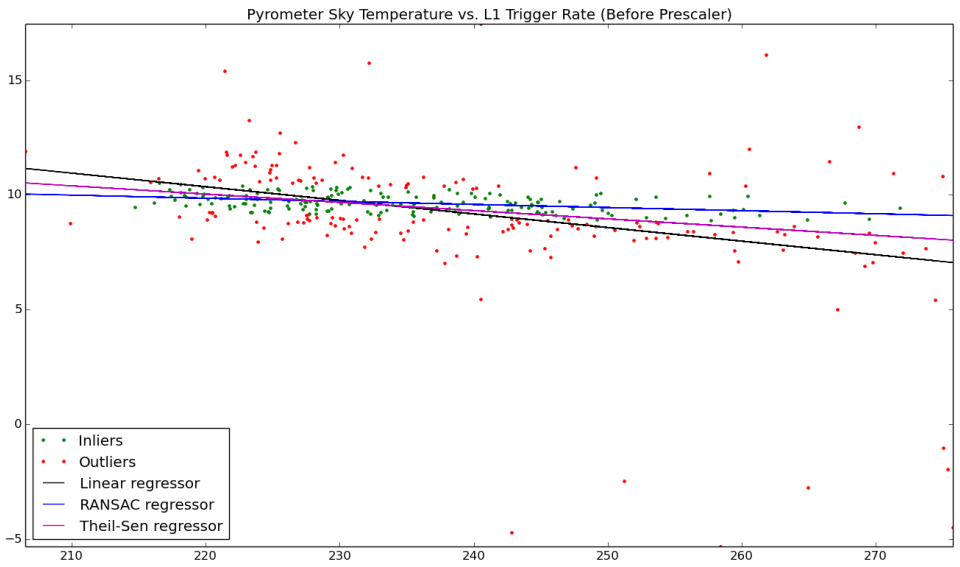

Figure 10.11: All the used estimators

The numbers from this estimation:

Estimator	Slope	Intercept
OLS	-0.059	23.403
Theil-Sen	-0.035	17.741
RANSAC	-0.027	15.982

R^2 OLS	0.146
R^2 Theil-Sen	0.122
R^2 RANSAC	0.0861

Table 10.5: Results of the robust regression

We can see that the results of the robust regression do not differ greatly from previously obtained results, using ordinary linear regression, however, from the plot it is obvious that the RANSAC algorithm is showing the greatest resilience to outliers.

Conclusion

In this thesis a web application was build and described into detail. The focus has been on making all the layers of the application using open-source technologies. Even though the application is fully functioning, it could benefit from a few more improvements and features. For example, user can not download the image of the plotted graph, which would be practical. Also, another idea for adding features for the application is the implementation of data analysis tools.

Starting with the data analysis, we have seen how data visualization is a very important step for getting a sense of the data structure. We described the data with plots and descriptive statistics, used histograms and box plots to summarize the parameters individually. This has proven to be very insightful in the sense that we have spotted numerous missing values in the data set, which we excluded from the analysis to avoid problems. Also, box plots have enabled us to notice that the trigger rate values contain some huge outliers, so we knew right away that this is a variable in need of transforming in the upcoming analysis. After we got a feel for the data, the statistical analysis follows. To find if there is a correlation and a relationship between variables, we used a simple linear regression, as the scatter plot showed the possibility of a linear relationship. The statistical measures of the relationship have showed that there is no significant linear relationship at all. However, knowing that the data set contains a lot of extreme values, we suspected that they are probably the ones responsible for such an outcome. Therefore, we resumed with our analysis. The output of the ordinary linear regression has showed very low R-squared value, of only 0.147. This means that there is a high degree of variability in the data, but that a certain trend can still be seen. The confidence interval is sufficiently narrow that we are able to conclude that the prediction from the model is satisfactory. Perhaps adding more variables to the model would help describe data better, however that is a dangerous game, because it is easy to fall into the trap of overfitting. In this case however, perhaps adding a related variable could improve the analysis, for example the sky brightness. Also, one could try to exclude the data deemed as outliers 'by hand', i.e. visually inspecting the scatter plot and excluding values grater/smaller than some value perceived as the desired threshold. We tried a different approach, excluding all the data that were three standard deviations from the mean, but that did not work and no data were excluded. Out of all methods of estimation, RANSAC method seems the most resilient in the presence of outliers, as seen from the plots. Based on the shape of the scatter plots and the regression fit line, I expected that the descriptive statistics will back the idea that there is a linear relationship between variables, but that has not been the case.

Appendix

The base-wide date change command

Write the following command into the terminal to fix all the dates in the database magicdb in the collection *dataset_ params*; dates are fixed from the string format to ISODate format.

```
mongo magicdb --eval db.dataset_params.find().forEach(function(doc)
        {doc.date = new ISODate(doc.date); db.dataset_params.save(doc);});
```

NB: This only needs to be done when importing the data from the .csv file, in which the date is in the format of the string. When date is in this format, it is not possible to search date ranges.

The shell script for the automatic starting of the magic_api.py service

```
#!/bin/bash

PIDFILE=/home/aprpic/magic/magic_api.pid
DAEMON=/home/aprpic/magic/magic_api.py
NAME="python /home/aprpic/magic/magic_api.py"

case $1 in
 start)
 echo -n "Starting magic-api..."
 start-stop-daemon --start --background --make-pidfile --pidfile
 $PIDFILE --exec $DAEMON --
 echo "OK"
 ;;
 stop)
 echo -n "Stoping magic-api..."
 start-stop-daemon --stop --name $NAME --pidfile $PIDFILE
 --retry=TERM/5/KILL/1 ||
 { echo "error"; exit 1;}
 echo "OK"
 ;;
 restart)
 ;;
 *)
 ;;
 esac
```

The code for the magic_api.py

```
#!/usr/bin/env python

import pymongo as mongo
from time import mktime
from datetime import datetime, timedelta
from bson.json_util import dumps, loads
from bottle import route, run, request, response, static_file, error

collection = mongo.Connection('localhost', 27017).magicdb.dataset_params

def get_params():
        d = datetime(2014, 12, 14)
        query_params = collection.find({"date": d, "telescope": "M1"},
                {'name':1,'_id':0})
        names = {r.get('name', '') for r in query_params}
        names.discard('')
        names.discard('NULL')
        return sorted(names)

param_names = get_params()

word_props = {'name', 'telescope'}


def iso_str_to_datetime(iso_date):
        return datetime.strptime(iso_date, "%Y-%m-%d")


def prepare_document_for_output(d):
        if 'date' in d:
                d['date'] = mktime(d['date'].timetuple()) * 1000
        return d

def asJSON(data):
        response.content_type = 'application/json; charset=utf8';
        return dumps(data)

@error(404)
def on_not_found(error):
    return static_file('404.html', root='./')

@route('/params')
```

```
43 def params():
44         return asJSON(param_names)
45
46 @route('/get')
47 def get():
48         # request.params when params are passed in URL (GET)
49         # request.forms when params are sent as data (POST)
50         # we support both ways by using one that is not empty
51         params = request.params or request.forms
52
53         query = {}
54         projection = {'_id': 0}
55
56         for name, value in params.iteritems():
57                 # special param that selects fields to return
58                 if name == 'output':
59                         for name in value.split(','):
60                                 projection[name] = 1
61                         continue
62
63                 if name == 'date':
64                         if '..' in value:
65                                 date_from, date_to = map(iso_str_to_datetime,
66                                         value.split('..'))
67                         else:
68                                 date_from = date_to = iso_str_to_datetime(value)
69
70                         value = {"$gte": date_from, "$lt": date_to +
71                                 timedelta(days=1)}
72
73
74                 elif name in word_props:
75                         if ',' in value:
76                                 value = {"$in" : value.split(',')}
77
78                 else: # numeric
79                         value = float(value)
80
81                 query[name] = value
82
83         cursor = collection.find(query, projection)
84         return asJSON([prepare_document_for_output(d) for d in cursor])
85
86
87
88 if __name__ == '__main__':
```

```
89          try:
90                  import _config
91          except ImportError:
92                  _config = object()
93
94          host = getattr(_config, 'host', 'magic-dev.fesb.hr')
95          port = getattr(_config, 'port', 8585)
96          debug = getattr(_config, 'debug', False)
97
98          run(server='paste', host=host, port=port, debug=debug)
```

The HTML file (index.html)

```
<!doctype html>

<html class="no-js" lang="">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

<title></title>

<meta name="description" content="">

<meta name="viewport" content="width=device-width,

initial-scale=1">



<link rel="stylesheet" href="css/bootstrap.min.css">

<link rel="stylesheet" href="css/bootstrap-theme.min.css">

<link rel="stylesheet" href="css/bootstrap-select.min.css">

<link rel="stylesheet" href="jquery-ui/jquery-ui.min.css">

<link rel="stylesheet" href="css/main.css">



<script src="js/vendor/modernizr-2.8.3.min.js"></script>

</head>

<body>

<!--[if lt IE 8]>

<p class="browserupgrade">You are using an

```

```
<strong>outdated</strong> browser. Please <a href="

http://browsehappy.com/">upgrade your browser</a> to improve

your experience.</p>

<![endif]-->


<div class="container">

<div class="container-fluid pull-left input-wrap">

<form id="input-form">

<div id="param1-inputrow" class="inputrow">

<fieldset>

<legend>Param X</legend>

<select name="param1-telescope" class="telescope-input

selectpicker">

<option selected>M1</option>

<option>M2</option>

</select>

<br />

<select name="param1-param" class="param-input selectpicker"

data-live-search="true">

<optgroup label="DAQ">

<option  value="calq_cal">Calibration runs: Calibration Charge

</option>

<option  value="calq_int">Interleaved runs: Calibration

```

```
89  Charge </option>
90
91  <option  value="calq_sig">Charge for Signal Events </option>
92
93  <option  value="bias_sig">Bias of Signal Extractor </option>
94
95  <option  value="hitfrac_sig">Hit Fraction of Signal Events
96
97  </option>
98
99  <option  value="arrtm_cal">Calibration runs: Arrival Time
100
101 </option>
102
103 <option  value="arrtm_int">Interleaved runs: Arrival Time
104
105 </option>
106
107 <option  value="arrtm_sig">Signal events: Arrival Time</option>
108
109 <option  value="arrtmrms_cal">Calibration runs: Arrival Time
110
111 RMS </option>
112
113 <option  value="arrtmrms_int">Interleaved runs: Arrival Time
114
115 RMS</option>
116
117 <option  value="arrtmrms_sig">Signal events: Arrival Time RMS
118
119 </option>
120
121 <option  value="ped_ped">Pedestal runs: Pedestal</option>
122
123 <option  value="ped_int">Interleaved runs: Pedestal</option>
124
125 <option  value="npe_int">Interleaved runs: Number of
126
127 Photoelectrons</option>
128
129 <option  value="pedrms_ped">Pedestal runs: Pedestal RMS</option>
130
131 <option  value="pedrms_int">Interleaved runs: Pedestal RMS
132
133 </option>
134
```

```
135 <option  value="cfact_int">Interleaved runs: C Factor</option>
136
137 <option  value="pix1041">Calibration laser monitoring diode,
138
139 pix1041</option>
140
141 </optgroup>
142
143 <optgroup label="Central control">
144
145 <option  value="drvzd">Zenith angle</option>
146
147 <option  value="drvdev_daq">Control Deviation of the Motors,
148
149 DAQ data</option>
150
151 <option  value="camhv_daq">Camera HV, DAQ data</option>
152
153 <option  value="camdc_daq">Camera DC, DAQ data</option>
154
155 <option  value="camdt_daq">Discriminator Threshold, DAQ data
156
157 </option>
158
159 <option  value="campd_daq">PD, DAQ data</option>
160
161 <option  value="campixtemp_daq">Pixel Temperature Camera, DAQ
162
163 data</option>
164
165 <option  value="meanpixtemp_daq">Pixel Temperature, DAQ data
166
167 </option>
168
169 <option  value="camclusttemp">(Non-zero) Cluster Temperature
170
171 </option>
172
173 <option  value="camvcelbias_daq">VCELs bias, DAQ data</option>
174
175 <option  value="camlv1temp">LV1 Temperature</option>
176
177 <option  value="camlv2temp">LV2 Temperature</option>
178
179 <option  value="camlv1hum">LV1 Humidity</option>
180
```

```
<option  value="camlv2hum">LV2 Humidity</option>

<option  value="camcoolfcptopleft">Camera Cooling: FCPTopLeft

Temperature</option>

<option  value="camcoolfcpbottright">Camera Cooling:

FCPBottRight Temperature</option>

<option  value="camcoolrcptopleft">Camera Cooling: RCPTopLeft

Temperature</option>

<option  value="camcoolrcpbottright">Camera Cooling:

RCPBottRight Temperature</option>

<option  value="camcoolchasiastopleft">Camera Cooling:

ChasiasTopLeft Temperature</option>

<option  value="camcoolchasiasbottright">Camera Cooling:

ChasiasBottRight Temperature</option>

<option  value="camcoolchasiasftopleft">Camera Cooling:

ChasiasFTopLeft Temperature</option>

<option  value="camcoolchasiasfbottright">Camera Cooling:

ChasiasFBottRight Temperature</option>

<option  value="camcoolrearbottleft">Camera Cooling:

RearBottLeft Humidity </option>

<option  value="camcoolreartopleft">Camera Cooling:

RearTopLeft Humidity</option>

<option  value="camcoolfrontbottright">Camera Cooling:

FrontBottRight Humidity </option>

```

```
227 <option  value="camcoolfronttopright">Camera Cooling:
228
229 FrontTopRight Humidity</option>
230
231 <option  value="amcerr">AMC Number of Panels in Error State
232
233 </option>
234
235 <option  value="l1t">L1 Trigger Rate (Before Prescaler)</option>
236
237 <option  value="l2t">L2 Trigger Rate (After Prescaler)</option>
238
239 <option  value="l2t_daq">L2 Trigger Rate (After Prescaler),
240
241 DAQ data</option>
242
243 <option  value="sumt_globr">Sum-Trigger-II Global Rate</option>
244
245 <option  value="sumt_l3">Sum-Trigger-II L3 Trigger Rate</option>
246
247 <option  value="sumt_dtw">Sum-Trigger-II Working Thresholds
248
249 </option>
250
251 <option  value="sumt_cbt1">Sum-Trigger-II Clip board
252
253 temperatures 1</option>
254
255 <option  value="sumt_cbt2">Sum-Trigger-II Clip board
256
257 temperatures 2</option>
258
259 <option  value="sumt_ac">Sum-Trigger-II: AC current
260
261 consumption</option>
262
263 <option  value="sumt_astrob">Sum-Trigger-II Astro-board
264
265 temperature</option>
266
267 <option  value="cool_crate">Cooling: VME fan trays temperature
268
269 </option>
270
271 <option  value="cool_rack">Cooling: Heat exchangers internal
272
```

```
273 temperatures</option>
274
275 <option  value="calbtemp1">Calibration Box Temperature</option>
276
277 <option  value="calbtemp2">Calibration Box Temperature, next
278
279 to heating plate</option>
280
281 <option  value="calbhum">Calibration Box Humidity</option>
282
283 <option  value="sg_devaz">Azimuthal Misspointing * Sin(Zenith)
284
285 </option>
286
287 <option  value="sg_devzd">Zenith Misspointing</option>
288
289 <option  value="sg_camcx">Camera Centre X</option>
290
291 <option  value="sg_camcy">Camera Centre Y</option>
292
293 <option  value="sg_stars">Number of Correlated Stars</option>
294
295 <option  value="sg_bright">Sky Brightness</option>
296
297 <option selected value="wea_temp">Atmospheric Temperature
298
299 </option>
300
301 <option  value="wea_hum">Atmospheric Humidity</option>
302
303 <option  value="wea_ws">Wind Speed</option>
304
305 <option  value="wea_gust">Wind Gusts</option>
306
307 <option  value="wea_see">Seeing (from TNG)</option>
308
309 <option  value="wea_dust">Dust concentration (from TNG)</option>
310
311 <option  value="rec_temp">Recievers Temperature</option>
312
313 <option  value="camtd_daq">Trigger Delay, DAQ data </option>
314
315 <option  value="camipr_daq">IPR, DAQ data</option>
316
317 <option  value="camiprerr_daq">IPR Error, DAQ data</option>
318
```

```
<option  value="pyro_cloud">Pyrometer, Cloudiness</option>

<option  value="pyro_skyt">Pyrometer, Sky Temperature</option>

<option  value="las_trans3km">Laser, Transmission from 3km

</option>

<option  value="las_trans6km">Laser, Transmission from 6km

</option>

<option  value="las_trans9km">Laser, Transmission from 9km

</option>

<option  value="las_trans12km">Laser, Transmission from 12km

</option>

<option  value="muon_psf">PSF from muons</option>

<option  value="muon_psfn">Number of muons for PSF calc</option>

<option  value="muon_size">Mean muon size</option>

<option  value="sbigpsf_b">PSF from SBig, B filter</option>

<option  value="sbigpsf_l">PSF from SBig, L filter</option>

</optgroup>

</select>

<div class="prop-selection-wrap">

<select name="param1-prop" class="prop-input selectpicker">

<option value="mean" selected>Mean</option>

<option value="median">Median</option>

<option value="max">Max</option>

<option value="min">Min</option>

```

```
<option value="rms">RMS</option>
</select>
<div class="checkbox">
<label for="param1-errors">
        <input type="checkbox" name="param1-errors" id=
"param1-errors" class="error-input" />
        Error bars (RMS values)
</label>
</div>
</div>
</fieldset>
</div>


<div id="param2-inputrow" class="inputrow">
<fieldset>
<legend>Param Y</legend>
<select name="param2-telescope" class="telescope-input
selectpicker">
<option selected>M1</option>
<option>M2</option>
</select>
<br />
<select name="param2-param" class="param-input selectpicker"
```

```
411 data-live-search="
412
413 <optgroup label="DAQ">
414
415 <option  value="calq_cal">Calibration runs: Calibration Charge
416
417 </option>
418
419 <option  value="calq_int">Interleaved runs: Calibration
420
421 Charge </option>
422
423 <option  value="calq_sig">Charge for Signal Events </option>
424
425 <option  value="bias_sig">Bias of Signal Extractor </option>
426
427 <option  value="hitfrac_sig">Hit Fraction of Signal Events
428
429 </option>
430
431 <option  value="arrtm_cal">Calibration runs: Arrival Time
432
433 </option>
434
435 <option  value="arrtm_int">Interleaved runs: Arrival Time
436
437 </option>
438
439 <option  value="arrtm_sig">Signal events: Arrival Time</option>
440
441 <option  value="arrtmrms_cal">Calibration runs: Arrival Time
442
443 RMS </option>
444
445 <option  value="arrtmrms_int">Interleaved runs: Arrival Time
446
447 RMS</option>
448
449 <option  value="arrtmrms_sig">Signal events: Arrival Time RMS
450
451 </option>
452
453 <option  value="ped_ped">Pedestal runs: Pedestal</option>
454
455 <option  value="ped_int">Interleaved runs: Pedestal</option>
456
```

```
<option  value="npe_int">Interleaved runs: Number of
Photoelectrons</option>
<option  value="pedrms_ped">Pedestal runs: Pedestal RMS</option>
<option  value="pedrms_int">Interleaved runs: Pedestal RMS
</option>
<option  value="cfact_int">Interleaved runs: C Factor</option>
<option  value="pix1041">Calibration laser monitoring diode,
pix1041</option>
</optgroup>
<optgroup label="Central control">
<option  value="drvzd">Zenith angle</option>
<option  value="drvdev_daq">Control Deviation of the Motors,
DAQ data</option>
<option  value="camhv_daq">Camera HV, DAQ data</option>
<option  value="camdc_daq">Camera DC, DAQ data</option>
<option  value="camdt_daq">Discriminator Threshold, DAQ data
</option>
<option  value="campd_daq">PD, DAQ data</option>
<option  value="campixtemp_daq">Pixel Temperature Camera, DAQ
data</option>
<option  value="meanpixtemp_daq">Pixel Temperature, DAQ data
</option>
<option  value="camclusttemp">(Non-zero) Cluster Temperature
```

```
503 </option>
504
505 <option  value="camvcelbias_daq">VCELs bias, DAQ data</option>
506
507 <option  value="camlv1temp">LV1 Temperature</option>
508
509 <option  value="camlv2temp">LV2 Temperature</option>
510
511 <option  value="camlv1hum">LV1 Humidity</option>
512
513 <option  value="camlv2hum">LV2 Humidity</option>
514
515 <option  value="camcoolfcptopleft">Camera Cooling: FCPTopLeft
516
517 Temperature</option>
518
519 <option  value="camcoolfcpbottright">Camera Cooling:
520
521 FCPBottRight Temperature</option>
522
523 <option  value="camcoolrcptopleft">Camera Cooling: RCPTopLeft
524
525 Temperature</option>
526
527 <option  value="camcoolrcpbottright">Camera Cooling:
528
529 RCPBottRight Temperature</option>
530
531 <option  value="camcoolchasiastopleft">Camera Cooling:
532
533 ChasiasTopLeft Temperature</option>
534
535 <option  value="camcoolchasiasbottright">Camera Cooling:
536
537 ChasiasBottRight Temperature</option>
538
539 <option  value="camcoolchasiasftopleft">Camera Cooling:
540
541 ChasiasFTopLeft Temperature</option>
542
543 <option  value="camcoolchasiasfbottright">Camera Cooling:
544
545 ChasiasFBottRight Temperature</option>
546
547 <option  value="camcoolrearbottleft">Camera Cooling:
548
```

```
549 RearBottLeft Humidity </option>
550
551 <option  value="camcoolreartopleft">Camera Cooling:
552
553 RearTopLeft Humidity</option>
554
555 <option  value="camcoolfrontbottright">Camera Cooling:
556
557 FrontBottRight Humidity </option>
558
559 <option  value="camcoolfronttopright">Camera Cooling:
560
561 FrontTopRight Humidity</option>
562
563 <option  value="amcerr">AMC Number of Panels in Error State
564
565 </option>
566
567 <option  value="l1t">L1 Trigger Rate (Before Prescaler)</option>
568
569 <option  value="l2t">L2 Trigger Rate (After Prescaler)</option>
570
571 <option  value="l2t_daq">L2 Trigger Rate (After Prescaler),
572
573 DAQ data</option>
574
575 <option  value="sumt_globr">Sum-Trigger-II Global Rate</option>
576
577 <option  value="sumt_l3">Sum-Trigger-II L3 Trigger Rate</option>
578
579 <option  value="sumt_dtw">Sum-Trigger-II Working Thresholds
580
581 </option>
582
583 <option  value="sumt_cbt1">Sum-Trigger-II Clip board
584
585 temperatures 1</option>
586
587 <option  value="sumt_cbt2">Sum-Trigger-II Clip board
588
589 temperatures 2</option>
590
591 <option  value="sumt_ac">Sum-Trigger-II: AC current
592
593 consumption</option>
594
```

```
595 <option  value="sumt_astrob">Sum-Trigger-II Astro-board
596
597 temperature</option>
598
599 <option  value="cool_crate">Cooling: VME fan trays temperature
600
601 </option>
602
603 <option  value="cool_rack">Cooling: Heat exchangers internal
604
605 temperatures</option>
606
607 <option  value="calbtemp1">Calibration Box Temperature</option>
608
609 <option  value="calbtemp2">Calibration Box Temperature, next
610
611 to heating plate</option>
612
613 <option  value="calbhum">Calibration Box Humidity</option>
614
615 <option  value="sg_devaz">Azimuthal Misspointing * Sin(Zenith)
616
617 </option>
618
619 <option  value="sg_devzd">Zenith Misspointing</option>
620
621 <option  value="sg_camcx">Camera Centre X</option>
622
623 <option  value="sg_camcy">Camera Centre Y</option>
624
625 <option  value="sg_stars">Number of Correlated Stars</option>
626
627 <option  value="sg_bright">Sky Brightness</option>
628
629 <option selected value="wea_temp">Atmospheric Temperature
630
631 </option>
632
633 <option  value="wea_hum">Atmospheric Humidity</option>
634
635 <option  value="wea_ws">Wind Speed</option>
636
637 <option  value="wea_gust">Wind Gusts</option>
638
639 <option  value="wea_see">Seeing (from TNG)</option>
640
```

```
641 <option  value="wea_dust">Dust concentration (from TNG)</option>
642
643 <option  value="rec_temp">Recievers Temperature</option>
644
645 <option  value="camtd_daq">Trigger Delay, DAQ data </option>
646
647 <option  value="camipr_daq">IPR, DAQ data</option>
648
649 <option  value="camiprerr_daq">IPR Error, DAQ data</option>
650
651 <option  value="pyro_cloud">Pyrometer, Cloudiness</option>
652
653 <option  value="pyro_skyt">Pyrometer, Sky Temperature</option>
654
655 <option  value="las_trans3km">Laser, Transmission from 3km
656
657 </option>
658
659 <option  value="las_trans6km">Laser, Transmission from 6km
660
661 </option>
662
663 <option  value="las_trans9km">Laser, Transmission from 9km
664
665 </option>
666
667 <option  value="las_trans12km">Laser, Transmission from 12km
668
669 </option>
670
671 <option  value="muon_psf">PSF from muons</option>
672
673 <option  value="muon_psfn">Number of muons for PSF calc</option>
674
675 <option  value="muon_size">Mean muon size</option>
676
677 <option  value="sbigpsf_b">PSF from SBig, B filter</option>
678
679 <option  value="sbigpsf_l">PSF from SBig, L filter</option>
680
681 </optgroup>
682
683 </select>
684
685 <div class="prop-selection-wrap">
686
```

```
687 <select name="param2-prop" class="prop-input selectpicker">
688
689 <option value="mean" selected>Mean</option>
690
691 <option value="median">Median</option>
692
693 <option value="max">Max</option>
694
695 <option value="min">Min</option>
696
697 <option value="rms">RMS</option>
698
699 </select>
700
701 <div class="checkbox">
702
703 <label for="param2-errors">
704
705         <input type="checkbox" name="param2-errors" id=
706
707 "param2-errors" class="error-input" />
708
709         Error bars (RMS values)
710
711 </label>
712
713 </div>
714
715 </div>
716
717 </fieldset>
718
719 </div>
720
721
722
723 <div>
724
725 <fieldset>
726
727 <legend>Date range</legend>
728
729
730
731 <div class="inputrow">
732
```

```
<label for="date-from">Date from:</label>

<div class="input-group">

<input id="date-from" name="date-from" type="test" class=

"datepicker form-control" placeholder="date from">

<span class="input-group-addon"><span class="glyphicon

glyphicon-calendar" /></span>

</div>

</div>

<div class="inputrow">

<label for="date-to">Date to:</label>

<div class="input-group">

<input id="date-to" name="date-to" type="test" class=

"datepicker form-control" placeholder="date to">

<span class="input-group-addon"><span class="glyphicon

glyphicon-calendar" /></span>

</div>

</div>

</fieldset>

</div>



<input type="submit" class="btn btn-block pull-left" value=

"Submit"></input>

</form>

```

```
779 </div>
780
781
782
783 <div id="output" class="pull-left">
784
785
786
787 </div>
788
789 </div>
790
791
792
793 <hr>
794
795
796
797 <footer>
798
799 <p>&copy; Company 2015</p>
800
801 </footer>
802
803 </div> <!-- /container -->
804
805 <script src=
806
807 "//ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"
808
809 ></script>
810
811 <script>window.jQuery || document.write('<script
812
813 src="js/vendor/jquery-1.11.2.min.js"><\/script>')</script>
814
815
816
817 <script src=
818
819 "//cdnjs.cloudflare.com/ajax/libs/dygraph/1.1.1/dygraph-combine
820
821 d.js"></script>
822
823 <script>window.Dygraph || document.write('<script
824
```

```
825 src="js/vendor/dygraph-combined.js"><\/script>')</script>
826
827
828
829 <script src="js/vendor/dygraph-synchronizer.js"></script>
830
831
832
833 <script src="jquery-ui/jquery-ui.min.js"></script>
834
835
836
837 <script src="js/vendor/bootstrap.min.js"></script>
838
839 <script src="js/vendor/bootstrap-select.js"></script>
840
841
842
843 <script src="js/main.js"></script>
844
845 </body>
846
847 </html>
```

The main JavaScript file (main.js)

```
(function($){

function getDefaultFormValues() {
        var dateTo = new Date(); // today (now)

        var dateFrom = new Date(dateTo); // today...
        dateFrom.setFullYear(dateFrom.getFullYear() - 1); // ...one year ago

        return {
                'date-from': dateToISO(dateFrom),
                'date-to':   dateToISO(dateTo),

                'param1-telescope': 'M1',
                'param1-param': 'wea_temp',
                'param1-prop': 'mean',

                'param2-telescope': 'M1',
                'param2-param': 'wea_temp',
                'param2-prop': 'mean'
        }
}

// date -> date string in yyyy-mm-dd format
function dateToISO(date) {
        return date.getUTCFullYear() + '-' + twoDigit(date.getUTCMonth() + 1) +
                S'-' + twoDigit(date.getUTCDate());
}

// 9 -> "09"
function twoDigit(n) {
        return ('0' + n).slice(-2);
}

// parse URI encoded parameters
function deparam(str) {
        var ret = {}, p, k, v, parts = str.replace(/\+/g, ' ').split("&");

        for (var i = 0, l = parts.length; i < l; ++i) {
                p = parts[i].split("=");
                k = decodeURIComponent( p[0] );
                v = decodeURIComponent( p[1] );
                ret[ k ] = v;
```

```
43          }
44
45          return ret;
46  }
47
48  var ignoreNextHashChange = false;
49  function handleHashChange() {
50          if (ignoreNextHashChange) {
51                  ignoreNextHashChange = false;
52                  return;
53          }
54
55          output('');
56          populateForm();
57          scrollTo('#input-form');
58  }
59
60  // fill form by hash or default values
61  function populateForm() {
62          var hash = location.href.split('#!')[1];
63          var params = hash ? deparam(hash) : getDefaultFormValues();
64          var inParams = Object.prototype.hasOwnProperty.bind(params);
65
66          $('#input-form :input[name]').each(function () {
67
68                  if (($(this).attr('type') === 'checkbox')) {
69                          $(this).prop('checked', inParams(this.name)
70                                  && params[this.name] === 'on');
71
72                  } else if (inParams(this.name)) {
73                          this.value = params[this.name];
74                  }
75          });
76  }
77
78
79  // function: doc -> doc[prop]
80  function propGetter(prop) {
81          return function (d) {
82                  // 0 -> null
83                  return d[prop] || null;
84          };
85  }
86
87  // propery-with-error getter
88  function propWithErrorGetter(prop) {
```

```
89          return function (d) {
90                  return d[prop] ? [d[prop], d.rms] : null;
91          };
92  }
93
94
95  // replace output
96  function output(x) {
97          return $('#output').empty().append(x);
98  }
99
100 // scroll to an element in document
101 function scrollTo(el) {
102         $('html, body').stop().animate({
103                 scrollTop: $(el).offset().top
104         }, 1000);
105 }
106
107 // output an error message
108 function error(msg) {
109         scrollTo(output($('<b />').css('color', 'red').text('ERROR: ' + msg)));
110 }
111
112 // Initialization of the document
113 $(function($){
114
115         populateForm();
116         window.onhashchange = handleHashChange;
117
118         // use bootstrap-select
119         $('.selectpicker').selectpicker();
120
121         // hide error-bars checkbox if param is not 'mean' nor 'median'
122         $('.prop-selection-wrap > select').change(function () {
123                 var $chk = $(this).parent().find('.checkbox');
124
125                 if (this.value === 'mean' || this.value === 'median') {
126                         $chk.show();
127                 } else {
128                         $chk.hide().find('input').prop('checked', false);
129                 }
130         }).change(); // run it now to init
131
132         // use jQuery UI datapicker
133         $('.datepicker').datepicker({
134                 changeMonth: true,
```

```
135                 changeYear: true,
136                 dateFormat: 'yy-mm-dd'
137     });
138 
139         $('#input-form').submit(onSubmit);
140 
141 });
142 
143 var pending; // pending requests
144 var paramXInput;
145 var paramYInput;
146 
147 function onSubmit(event) {
148         event.preventDefault();
149 
150         ignoreNextHashChange = true;
151         location.hash = '#!' + $('#input-form').serialize();
152 
153         if (pending) {
154                 pending.forEach(abort);
155                 pending = null;
156         }
157 
158         var isoDateFrom = $('#date-from').val();
159         var isoDateTo = $('#date-to').val();
160 
161         if (!isoDateFrom || !isoDateTo) {
162                 error('date range not defined');
163                 return;
164         }
165 
166         if (isoDateFrom > isoDateTo) {
167                 error('invalid date range');
168                 return;
169         }
170 
171         var commonInput = {
172                 dateFrom: isoDateFrom,
173                 dateTo: isoDateTo
174         };
175 
176         paramXInput = $.extend(getParamInput('#param1-inputrow'), commonInput);
177         paramYInput = $.extend(getParamInput('#param2-inputrow'), commonInput);
178 
179         pending = [
180                 getParamData(paramXInput),
```

```
181                 getParamData(paramYInput)
182         ];
183
184         $.when.apply($, pending)
185         .done(onResults)
186         .fail(function (x) {
187                 error('' + x.status);
188         });
189
190         scrollTo(output('Loading...'));
191
192 }
193
194 function abort(x) {
195         x.abort();
196 }
197
198 // get input values for a param
199 // element -> object
200 function getParamInput(x) {
201         var $div = $(x);
202         var $param = $div.find('select.param-input');
203         var $prop  = $div.find('select.prop-input');
204
205         return {
206                 telescope: $div.find('select.telescope-input').val(),
207                 param: $param.val(),
208                 paramDesc: getSelectedDesc($param),
209                 prop: $prop.val(),
210                 propDesc: getSelectedDesc($prop),
211                 errorBars: $div.find('input.error-input').is(':checked')
212         };
213 }
214
215 // get selected option text
216 function getSelectedDesc(el) {
217         var $el = $(el);
218         var $option = $el.find('option[value="'+$el.val()+'"]');
219         return $.trim($option.text());
220 }
221
222 // make JSON request for a param
223 function getParamData(input) {
224         return $.getJSON('api/get', {
225                 date: input.dateFrom + '..' + input.dateTo,
226                 telescope: input.telescope,
```

```
227                 name: input.param,
228                 output: 'date,' + input.prop + (input.errorBars ? ',rms' : '')
229         });
230 }
231
232 // handle success of JSON requests
233 function onResults(paramXArgs, paramYArgs) {
234
235         var paramXResults = paramXArgs[0];
236         var paramYResults = paramYArgs[0];
237
238         output('Sorting...');
239         paramXResults.sort(compareByDate);
240         paramYResults.sort(compareByDate);
241
242         output('Building...');
243
244         var graphDivs = [];
245
246         function createGraphDiv() {
247                 var div = $('<div class="graph-wrap" />')[0];
248                 graphDivs.push(div);
249                 return div;
250         }
251
252         createXYGraph(createGraphDiv(), paramXInput, paramXResults, paramYInput,
253                 paramYResults);
254
255         var timeGraphs = [
256                 createTimeGraph(createGraphDiv(), paramXInput, paramXResults),
257                 createTimeGraph(createGraphDiv(), paramYInput, paramYResults)
258         ];
259
260         output(graphDivs).append(
261                 $('<a class="btn"><span class="glyphicon glyphicon-search"
262                         aria-hidden="true"></span> Raw data</a>')
263                         .click(function (event) {
264                                 event.preventDefault();
265                                 showRawData(paramXResults, paramYResults);
266                         })
267         );
268
269         if (timeGraphs.length > 0) {
270                 var dateWindow = getCommonDateWindow(timeGraphs);
271                 Dygraph.synchronize(timeGraphs);
272                 timeGraphs[0].updateOptions({dateWindow: dateWindow,
```

```
273                             isZoomedIgnoreProgrammaticZoom: true});
274         }
275
276 }
277
278 // get window range to include data of given graphs
279 function getCommonDateWindow(graphs) {
280         var left = Infinity;
281         var right = -Infinity;
282         graphs.forEach(function (g) {
283                 var range = g.xAxisRange();
284                 left = Math.min(left, range[0]);
285                 right = Math.max(right, range[1]);
286         });
287         return [left, right];
288 }
289
290 // param-input -> label
291 function labelFromInput(input) {
292         return input.paramDesc + ' ' + input.prop + ' (' + input.telescope + ')';
293 }
294
295 function createTimeGraph(div, input, data) {
296         var toY = input.errorBars ? propWithErrorGetter(input.prop) :
297                  propGetter(input.prop);
298
299         data = data.map(function (d) {
300                 return [ new Date(d.date), toY(d) ];
301         });
302
303         var label = labelFromInput(input);
304
305         return new Dygraph(div, data, {
306                 width: 700,
307                 height: 300,
308                 title: label + ' vs Time',
309                 labels: ['Time', label],
310                 legend: 'always',
311                 showRoller: true,
312                 errorBars: input.errorBars
313         });
314 }
315
316 function createXYGraph(div, inputX, dataX, inputY, dataY) {
317         var toY = inputY.errorBars ? propWithErrorGetter(inputY.prop) :
318                  propGetter(inputY.prop);
```

```
319         var toX = propGetter(inputX.prop);
320
321         var data = joinData(dataX, dataY, compareByDate, function (dx, dy) {
322                 var x = toX(dx);
323                 var y = toY(dy);
324                 return (x != null && y != null) ? [x, y] : null;
325         });
326         data.sort(compareByX);
327
328         var xlabel = labelFromInput(inputX);
329         var ylabel = labelFromInput(inputY);
330
331         return new Dygraph(div, data, {
332                 width: 700,
333                 height: 300,
334                 title: ylabel + ' vs ' + xlabel,
335                 labels: [xlabel, ylabel],
336                 xlabel: xlabel,
337                 ylabel: ylabel,
338                 strokeWidth: 0,
339                 drawPoints: true,
340                 legend: 'always',
341                 showRoller: true,
342                 errorBars: inputY.errorBars
343         });
344 }
345
346 // generate row data and display it in a new window
347 function showRawData(dataX, dataY) {
348         var xprop = paramXInput.prop;
349         var yprop = paramYInput.prop;
350         var xerror = paramXInput.errorBars;
351         var yerror = paramYInput.errorBars;
352
353         var lines = joinData(dataX, dataY, compareByDate, function (dx, dy) {
354                 var isoDate = dateToISO(new Date(dx.date));
355                 return isoDate + ' ' + dx[xprop] + (xerror ? ' '+dx.rms : '') + ' '
356                         + dy[yprop] + (yerror ? ' '+dy.rms : '');
357         });
358
359         var rawData = lines.join('\n');
360
361         window.open('', 'Raw data').document.write('<pre>'+rawData+'</pre>');
362 }
363
364 function joinData(X, Y, compare, getRow) {
```

```
365         var rows = [];
366         var i = 0, j = 0;
367         var lenX = X.length;
368         var lenY = Y.length;
369         if (!lenX) return rows;
370 
371         while (i < lenX && j < lenY) {
372                 var cmp = compare(X[i], Y[j]);
373 
374                 if (cmp < 0) {
375                         ++i;
376 
377                 } else if (cmp > 0) {
378                         ++j;
379 
380                 } else {
381                         var row = getRow(X[i], Y[j]);
382                         if (row != null) rows.push(row);
383                         ++i;
384                         ++j;
385                 }
386         }
387 
388         return rows;
389 }
390 
391 function compareByDate(a, b) {
392         return a.date - b.date;
393 }
394 
395 function compareByX(a, b) {
396         return a[0] - b[0];
397 }
398 
399 })(jQuery);
```

Bibliography

[1] E. Lorenz et al., A&G, Volume 46, Issue 6 (2005) 21-25

[2] B.S.Acharya et al., Astroparticle Physics 43 (2013) 3-18

[3] https://portal.cta-observatory.org/Pages/Home.aspx

[4] T.C.Weeks, *Very High Energy Gamma-Ray Astronomy*, page 4

[5] http://community.dur.ac.uk/ dph0www4/ground.php

[6] https://ned.ipac.caltech.edu/level5/Sept09/Cui/Figures/figure1.jpg

[7] F. Aharonian et al., 2008, Rep. Prog. Phys. 71, Number 9

[8] Rene A. Ong, VERY HIGH-ENERGY GAMMA-RAY ASTRONOMY, UCLA, http://www.astro.ucla.edu/ rene/talks/Rencontres-Ong-Writeup.pdf

[9] *ftp://ftp.cordis.europa.eu/pub/esfri/docs/esfri_roadmap_update_2008.pdf*

[10] https://magic.mpp.mpg.de/newcomers/cherenkov-telescopes/

[11] The CTA Consortium, Exp Astron (2011) 32:193–316

[12] The Next Generation of Cherenkov Telescopes. A White Paper for INAF

[13] https://en.wikipedia.org/wiki/Photomultiplier

[14] *Study of optical properties of last generation photodetectors for Cherenkov astronomy applications* M.E.Trias, Thesis

[15] M.E.Trias, *Thesis: Study of optical properties of last generation photodetectors for Cherenkov astronomy applications*, 2006.

[16] Barrio et al., 1998.

[17] J.Cortina et al., 2009., *Technical Performance of the MAGIC Telescope*

[18] http://www.ipp.phys.ethz.ch/research/magic_telescope

[19] Monitoring and calibration of the atmosphere in MAGIC, ICRC 2013, Ll. Font et al.

[20] J.Buckley et al., (Whipple Collaboration), 2008 Astron. Astrophys.329, 639

[21] J.Cortina,*Status and first results of the magic telescope*, Astrophysics and Space Science 297: 245-255, 2005.

[22] Ka Lok Cheng, Gustavo E. Romero, *Multiwavelength Approach to Unindentified Gamma-Ray Scources*, page 247

[23] A.M. Hillas, Astroparticle Physics 43 (2013) 19-43

[24] F.Zappa et al., Sens. Actuators A, vol. 140, pp. 103-112, 2007

[25] http://www.isgtw.org/feature/grand-vision-cherenkov-telescope-array

[26] http://www.acunetix.com/websitesecurity/web-applications/

[27] https://www.scriptrock.com/articles/mysql-vs-mongodb

[28] http://docs.mongodb.org/manual/core/cursors/

[29] http://www.bogotobogo.com/python/MongoDB_PyMongo/python_MongoDB_pyMongo_tutorial_installing.php

[30] http://bottlepy.org/docs/dev/index.html

[31] http://bsonspec.org/

[32] https://jqueryui.com/

[33] https://jqueryui.com/datepicker/

[34] https://upload.wikimedia.org/wikipedia/commons/thumb/8/8e/Histogram_example.svg/2000px-Histogram_example.svg.png

[35] https://researchutopia.files.wordpress.com/2012/11/boxplot1.png?w=580&h=299

[36] http://stattrek.com/statistics/charts/boxplot.aspx?Tutorial=AP

[37] http://www.itl.nist.gov/div898/handbook/eda/section3/eda35.htm

[38] http://nbviewer.ipython.org/github/justmarkham/DAT4/blob/master/notebooks/08_linear_regression.ipynb