Effect of Atmospheric Pressure on Cosmic Raysy

Michael McEllin

1 Background and Motivation

We know that a single high-energy cosmic ray particle, often a proton, can initiate a cascade of particle collisions in the atmosphere, leading to an Extended Air Shower (or 'EAS').

An EAS cascade cannot occur in the absence of the atmosphere, but the particles generated in the cascade are also absorbed by the atmosphere. If we could place a series of cosmic ray detectors at different altitudes along the path of an EAS, at very high altitudes we would expect to see very few particles (but individually they would be of very high energy) and as we come down through the atmosphere the number of particles would increase, but the individual energies would decrease as energy is shared out. In the lower layers of the atmosphere, however, the individual particle energies drop below the threshold level required to create new particles. From that point onwards, more and more of the remaining particles are absorbed by collisions with air molecules.

Ground based cosmic ray detectors, such as those in the HiSPAC network, are in the lower part of the atmosphere where new particles are not being created and many are being absorbed. The less energetic EAS events may not trigger the detectors at all, because too many particles have already been blocked by the atmosphere.

We might suspect that the development of a cosmic ray shower depends principally on the amount of matter encountered during the passage of the shower through the atmosphere. We might therefore expect that on days when the atmospheric pressure is higher (that is, there is a greater mass of air between us and space) we would see a lower rate of EASs at ground level (more showers have been completely absorbed). In contrast, low atmospheric pressure means that the peak of cascades occurs lower in the atmosphere, giving a greater intensity of cosmic rays at ground level. We might also expect that tracks that pass through more air (e.g. those at a

larger angle to the vertical) undergo a more development, and ultimately more absorption.

We also need to worry about the effects of the environment on the efficiency of the detectors. Perhaps, for example, the light-intensifier tubes are sensitive to temperatures and produce bigger signals at lower temperatures. (Their specification suggests that this may be so.) In this situation it would, of course, be temperature inside the detector 'top-box', that matters rather than the air temperature outside the box. When the box is exposed to direct sunlight the inside temperature may well be considerably higher than the outside temperature. It is therefore entirely plausible that we might register differing rates EAS events depending on the temperature (even though the real event rate is not changing).

The cosmic ray event rate may, perhaps, also be correlated with other factors, such as, for example, the intensity of the solar wind. The solar wind is a stream of hot ionised gas blowing outwards from the Sun. Being ionised it is conducting, and being conducting it will carry currents and a magnetic field, and magnetic fields can deflect the path of charged particles.

The magnetic fields can be strong enough to deflect the lower-energy cosmic ray particles coming from outside the Solar System and stop them arriving at the Earth. So, when the wind is strong we might expect to get more protection and see a lower rate of EAS events. The solar wind tends to be strongest when there are lots of sunspot on the solar surface, and such sunspots are also the source of the magnetic fields that get trapped in the streaming gas. On the other hand, giant solar flares, which are often associated with sunspots may themselves generate lower energy cosmic rays and these will stream towards the Earth, sometimes adding to the cosmic ray flux.

It is by no means certain that we would be able to easily see such effects with the HiSPARC detectors. They are most sensitive to cosmic rays which arrive at the upper atmosphere with energies of about $10^{15}eV$ and these are not greatly affected by the Solar Wind.

All of these and other hypothetical sources of variation in the cosmic ray intensity would, however, be masked by the relatively strong effects of variations in atmospheric pressure. In order to look for such correlations it would help if we could remove the effects of pressure to obtain a 'pressure-corrected' EAS event rate in which remaining variations ought to be more apparent.

The approach used by professional scientists in situations such as this is to first identify the strongest effect on the observed data. They then use any observed correlation to remove this from the observations. So, for

example, the event rate on high-pressure days would be adjusted upwards and conversely would be adjusted downwards on low-pressure days, such that we would no longer be able to detect any pressure correlations in the adjusted data. We then look for the next strongest source of variation, perhaps temperature, and remove this, and so on. This is known as *Analysis of Variance'*(usually abbreviated to ANOVA). The process of deriving the influence of a particular source of variation usually involves a statistical process called *Regression*. We will talk about this below.

The advantage of this approach is that we know when we have reached the end, and can be certain that there are no remaining significant correlated effects on the data. We can be certain of this because when the data is fundamentally random (which we believe to be the case with cosmic ray arrival rate) there will always be a inherent level of variation in the arrival rate. If the long term average arrival rate were, say, 2500 events per hour, the actual counts in any particular hour is very unlikely to be exactly 2500, it will always be a little above or below this figure. In fact, we can do the mathematics and predict exactly how likely it is that we might see any particular arrival count in any particular hour: it is called the Poisson Distribution, named after a great French mathematician. Hence, once we see that the distribution of arrival rates, after correction for known correlations, looks like the Poisson Distribution corresponding to the long term average rate we can be sure that there are no other significant hidden correlations. (There may, of course, still be some weak correlations, too insignificant to show up.)

I like to look at things from this viewpoint: the first stage in any physics experiment is understanding and calibrating the instruments that you intend to use. We need to know that we can trust what they tell us. In the case of high energy cosmic ray detection, the ground-based detectors are in effect only part of the complete detector: we do, indeed, rely on the interaction with the atmosphere to create the EAS which spreads the energy of the initiating particle over a wide area and increases the probability of interactive with a fixed detector (We would have a much lower chance of having our detector in the right place to register a single proton). We must therefore calibrate the behaviour of the atmospheric component of our detector.

2 Project Learning Outcomes

- Insight into why using very large amounts of data allows us to tackle different types of scientific problem
- The use of statistical methods, particularly *Regression* and *Analysis of Variance*.
- Introduction to some of the programming techniques required for handling large amounts of data.
- Practice in methods of presenting information extracted from large datasets.

3 The Project

We can just plot the rate of cosmic ray events and against pressure and we would certainly find a correlation. This is not, however, getting to heart of the physical processes.

We have no particular reason at present to believe that in space outside the Earth's atmosphere that there is any preferred direction from which cosmic rays arrive[1]. Here on Earth, in the absence of the atmosphere, if we plotted the arrival directions on the sky over some period we would expect to see a fairly uniform scatter.

Once we start considering the effect of the atmosphere, however, we must take account of the different lengths of the cosmic ray shower tracks through the atmosphere. Those that arrive from directly overhead pass through the smallest amount of atmosphere, while those coming from nearer the horizon have longer tracks within the atmosphere and so interact with a larger number of air molecules. Our crude plot of EAS rate against pressure is therefore merging together the data from tracks that see very different amounts of air before they reach our detectors. To get at the fundamental physical processes it would be better if we could correlate the rate of arrival of cosmic rays against the amount of air encountered along each track.

We can do this because there is some HiSPARC data available showing the directions from which showers arrive. We can, in principle, use the angle

[1] A great deal of effort has gone into trying to detect preferred directions looking for a clues to the origin of cosmic rays. It is possible that at the very highest energies there may be some correlation with the direction of super-massive black holes in other galaxies—so-called 'active galactic nuclei' or AGNs. In the range of energies accessible to the HiSPARC network any correlations are likely to be undetectable.

or arrival of a shower and the current barometric pressure to assign an air mass to each track. That is the essence of this project.

We usually describe the direction of an EAS arrival in terms of the *zenith angle* θ, which is the angle with the line going from the detector straight upwards to the point on the sky known as the *zenith*, and an *azimuth angle*, ϕ which measures the projection of the track down onto the ground—effectively a compass bearing.

For *this* project, however, we only need to pay attention to the zenith angle, so we can ignore the azimuth angle[2].

There are, in principle, two ways to get EAS arrival directions, but I suggest that for the present purposes we confine ourselves to direction information registered by those HiSPARC stations that contain four detector plates. (mostly those at universities in the Netherlands, such as the Nikhef group). For these we know that the accuracy of the direction information has been calibrated against data from professional cosmic ray observatories. We can rely on it without further consideration[3].

Showers arriving from high zenith angles clearly pass through more air mass. For example, those coming from, say 45^o to the zenith have passed through $\sqrt{2}$ times as much air, compared to those coming straight down (zenith angle of zero). (That is, we have to divide by $cos(\theta) = cos(\pi/4) = \frac{1}{\sqrt{2}})$[4] Alternatively, we would have to raise the air pressure by $\sqrt{2}$ to get the same amount of air between us and space directly overhead. A shower arriving at 60^o to the zenith passes through twice as much air mass. (That is, we have to divide by $cos(\theta) = cos(\pi/3) = \frac{1}{2}$).

This suggests that rather than plotting total EAS event rate against pressure, we should sort into zenith angle groups and plot the density of tracks on the sky against an estimate of the air mass per unit area along the track which we obtain by dividing the barometric pressure by the cosine of the zenith angle.

---

[2] The azimuth angle may be conventionally measured from either the direction of North or the direction of South, and in general you need to know which of these conventions are in use. HiSPARC uses the North convention, but most other astronomers use South. The convention is, however, not relevant for the current work.

[3] From work we did a few years ago at Marling, I have some doubts about the accuracy of the other method, which compares EAS arrival times at multiple stations. I think that there may be systematic errors in this method which we were unable to quantify.

[4] Note that I have started describing angles in terms of *Radians* (in which the angle all the way round a circle is 2π radians) rather than degrees. You need to get use to this for programming with trigonometry because all the mathematical functions of programming languages need you to supply angles as radians. Convert from degrees to radians by multiplying by $\pi/180$.

The air mass per unit area traversed will be our independent variable. Our dependent variable should be the *density* of shower tracks at a particular zenith angle. That is, the number of showers per hour per unit of *solid angle*[5]. Imagine plotting all the showers on a sphere. It is the density per unit area on this sphere. We therefore have another correction to make before we do our plots because there is a lot more sky at larger zenith angles than there is at smaller angles. Think of the amount of sky just above the horizon, all the way round, compared to the tiny area just around the zenith.

I am just going to quote a formula here, because the proper derivation of the correction requires A-level calculus. (See the Appendix.) For tracks with zenith angles between θ_1 and θ_2 in order to get the number of tracks *per steradian* (which is the unit for solid angles—areas on the sky) we divide the EAS rate count by a factor, f :

$$f = \pi(sin^2(\theta_2) - sin^2(\theta_1)) \tag{1}$$

This clearly has the right sort of form, because it goes to zero at the zenith (where there is an infinitesimal amount of sky) and also goes to zero at the horizon (where there is zero projected detector plate area) and is a maximum half-way between.

3.0.1 Calculation Work

In order to work out how we get to the data that we can plot for our correlation it is a good idea to start from the end.

In order to plot our final graph we need to have a table with two essential columns (those which are starred), but it is probably a good idea to include the other columns for checking that our results make sense.

mass/unit area along track* This we calculate by taking the barometric pressure and dividing by the cosine of the zenith angle. This is our independent variable in graph plotting and curve fitting.

counts/steradian* The count rate at a particular zenith angle logged over some period of time. This is the actual counts we record in the zenith angle interval divided by the factor given by equation 1. This will be the dependent variable in graph plotting and curve fitting.

[5] The unit of solid angle is the *steradian*. There are 4π steradians all around a sphere (or 2π in a hemisphere). You therefore get a measure in steradians if you divide an area on a sphere of unit radius by 4π.

time period logged Assuming that we are logging over, say, hourly intervals this might be the start of the logging period.

zenith angle The zenith angle which we may take to be the lower boundary of a zenith angle range (e.g. we write 60^o for the interval from 60^o to 70^o).

That is where we need to end up. Note that we might choose to write this data to a disk file, perhaps in tab separated columns, but before we do that we would be holding this information in several internal data arrays in our analysis program and we might choose to plot directly from there (e.g. using the *matplotlib* library in Python) with going through a file.

Our main calculation challenge is that the raw data from the HiSPARC data base will consist of two tables organised in quite different ways. (You can download this data using the form on the HiSPARC website at https://data.hisparc.nl/data/download/. I suggest that you select the station *Nikhef 501* and download a small amount of data—say one or two hours—to start with. It is easy to test programs using a small amount of data, and when we are sure it is working we then move on to working with several months worth of data.)

The first table records EAS events and it has the following columns, in this order:

date time of event [GPS calendar date];

time time of event [GPS time of day];

timestamp time of event [UNIX timestamp];

nanoseconds time of event [number of nanoseconds after timestamp];

pulse heights (4x) maximum signal pulse height [ADC]—four columns of these;

integral (4x) integral of the signal [ADC.sample]—four columns of these;

number of mips (4x) estimate for the number of particles in the detector—four columns of these;

arrival times (4x) relative time of arrival of the first particle in the detector [ns]—four columns of these;

trigger time relative time of the trigger timestamp [ns]

zenith reconstructed shower axis zenith angle [deg]. N.B. if it was not possible to calculate angle data for the event (most of them!) then this value will be set to -999. Reconstruction of the arrival track only occurs if the shower triggers all four detector plates.

azimuth reconstructed shower axis azimuth angle [deg]

We are really only interested for this project in the time columns and the zenith column. (We will find that the *timestamp* column is the most useful measure of time since it just counts seconds and we do not have to worry about how-many-days-in-a-month and so on.)

If you want to know what the rest mean, look up the documentation on the HiSPARC website.

Note that there is one line (or 'record') for every EAS event logged. Since these occur at random times if we want to know the rate at which events occur (say, per hour) we will have to count the number of events in a period of 3600 seconds. We will be interested only in events that have a valid zenith angle calculation (i.e. its value will NOT be set to -999). Data for all other events should be ignored.

The weather data table has these columns:

date time of event [GPS calendar date]

time time of event [GPS time of day]

timestamp time of event [UNIX timestamp]

temperature inside temperature inside [deg C]

temperature outside temperature outside [deg C]

humidity inside relative humidity of the air inside [%]

humidity outside relative humidity of the air [%]

atmospheric pressure barometer data [hPa]

wind direction wind direction [deg]

wind speed wind speed [m/s]

solar radiation intensity of solar radiation [W/m/m]

uv index measurement for UV intensity [0-16]

evapotranspiration amount of evaporated and transpirated water [mm]

rain rate amount of rainfall [mm/h]

heat index perceived temperature taking humidity into account [deg C]

dew point water vapor below this temperature will start to condensate [deg C]

wind chill perceived temperature taking wind into account [deg C]

Once again, we only need to pay attention to part of this data, the time, the temperature (outside) and the atmospheric pressure. Note that, unlike the event data, the weather data is logged at regular intervals, typically every five seconds.

The calculations we need to do therefore run something like this:

1. Read the weather data from its file, line by line.
   (a) Divide the data into hourly intervals (or half-hourly if you wish).
   (b) Calculate the average atmospheric pressure and temperature over that hour.
   (c) Build three internal data arrays (e.g. appending data to the arrays hour by hour):
      i. The timestamp at the start of the following hour.
      ii. The average pressure during the following hour.
      iii. The average temperature during the following hour.
2. Read the event data from the file line by line, until we reach the end of the file:
   (a) Check record for valid zenith angle data. IF invalid read the next record from the event data file.
   (b) For a record with valid zenith angle, extract the timestamp value (3rd column) and the zenith angle (column 22?) and append these to an internal pair of arrays one holding times the other zenith angles.

In each hour we now need to count up the cosmic ray event that occur in each zenith angle band, ending up with a table something like the rough example in Table 1.

Table 1: Example Table for hourly counts against zenith angle for each hour

Hour	**Press**	$0 \rightarrow 10^o$	$0 \rightarrow 10^o$	$10 \rightarrow 20^o$	$20 \rightarrow 30^o$	$30 \rightarrow \ldots$
1	1005	30	35	27	etc.	etc. ...
2	998	32	28	46	etc .	etc. ...
3	988	etc	etc	etc	etc .	etc. ...

You continue doing this and adding line to the table, until you have worked through every hour for which data has been logged.

There are several ways you can program this counting process. I might, for example, set up an iteration loop on the weather data, working out the start and end of the hourly periods in terms of the timestamp values (add 3600 to go from start to end). We then start at the beginning of the event data table and assign the counts to the appropriate zenith angle box in the first row, until the timestamp on an event becomes larger than the hour-end time.

You will, of course, need to set up arrays to hold the tabular data that you are going the generate. Python then has the very convenient **append()** operation to add a new item onto the end of an array. (There are faster ways to do this, but keep it simple to start with.)

This is still not the table that relates EAS event rate to air mass along the arrival track.

1. We now iterate through lines in Table 1
    (a) For each line we then iterate through the zenith angles for each angle creating a new record in our final output table containing:
        i. An airmass/unit area calculated by dividing the pressure by the cosine of the zenith angle.
        ii. The EAS event rate/steradian by dividing the event rate in Table 1 by the scaling factor from equation 1

4 Some Technical Background

4.1 Understanding Data Mining

'Big Data' is one of the buzz-words of the moment—and it really is more than just hype.

In truth, the modern terms *Data Mining*, *Data Analytics* or *Data Science* are to a large extent just convenient labels applied to a collection of techniques we have used on smaller scales for many years. More organisations now required these skills to be applied on larger scales and applied together in systematic ways, and the demand from industry means that many universities now offer masters courses that teach these techniques to prepare students for a thriving job market.

Modern computing technology allows us to collect and store information at rates and volumes that were never before possible. Many areas of science are now exploiting this capability—and having to learn how to deal with data mountains. It opens the prospects of new types of experiment, with the chance of observing previously hidden processes, but also demands that scientists learn new techniques.

Commercial organisations have similar issues: every time you buy from Amazon, it helps them to build a profile of the type of things people like you tend to buy, and often specifically what you as an individual are likely to buy, so they can persuade you to buy more. Many companies now explicitly advertise for 'Data Scientists' to help handle their data mountains.

For example:

- Experiments on the Large Hadron Collider (which have been described as 'looking for very small needles in very large haystacks') have to handle data coming out of its detectors at a rate equivalent to the entire information flow on the Internet.

- Automated genome decoding is now undertaken by rooms full of robotic DNA sequencers, followed by sophisticated statistical analysis on supercomputers to put together a complete genome sequence.

- The recent image of a black hole from the Event Horizon Telescope involved processing hundreds of terabytes of data collected over just a couple of days.

- Sometime in the next couple of years, the new *Large Synoptic Survey Telescope*, will be completed and by 2022 its automated observation programme—scanning the entire sky visible from its site every few

days—will be matching or exceeding the LHC data rates. (The romance of astronomers going to remote mountain peaks to spend cold nights sitting at the end of a telescope—which still happened when I was a student—now belongs to times past: we now control telescopes robotically to do the actual observations and human astronomers sit at computer monitors half a world away.)

- A few years from now the *Square Kilometre Array*—a new radio telescope of extraordinary sensitivity—will start producing data at rates that exceed by an order of magnitude everything we have just talked about.

In all of these fields—and more—much of the research is about write programs that manipulate data stored in a computer, *mining* (and filtering) to extract useful information. In effect, the experiments are 'virtual' rather than involving real equipment in laboratories.

Scientists and mathematicians who understand how to process and filter large amounts of data are now in high demand not just in the fields highlighted above, but also in commercial organisations whose competitive position depends on the effective deployment of the large quantity of information flowing into company computers every day. Before I retired, I used to monitor the behaviour of nuclear reactors using data collected minute by minute, and sometimes we needed to think about what might happen to the reactors years into the future (for example, if we were considering changing the amount of uranium in the fuel). These studies, using large and complicated software that simulated the reactor physics, generated tens to hundreds of gigabytes of data which had to be filtered to reveal the answers for which we were looking.

Hence, specialists in High Energy Physics and Astronomy are now sometimes poached away to high-paying commercial jobs, such as in City financial institutions.

The HiSPARC experiment is a small example of 'Big Data'. There is, in fact, relatively little to be learned from just one Extended Air Shower event, but a great deal may be possible by observing patterns that emerge over long periods of time from many HiSPARC stations. While it is, indeed, possible to detect some interesting effects by downloading small amounts of information from the HiSPARC database (say 7-14 days worth) and analysing these using, say, a spread sheet, the essentially random nature of cosmic ray events means that precision measurements require handling information collected over months and years.

A HISPARC station detects, on average, between 1500 and 3000 EAS events per hour[6]—up to, say, 60,000 per day, or getting on for 20,000,000 per year. That is both exciting and daunting: that volume of material means that statistical analysis could be very successful at measuring small effects, but we also need to learn how to efficiently process tables with 20 million entries. (And remember, the HiSPARC database is small compared to examples such as the LHC.)

Nevertheless, we will need to learn some of the techniques that are now required to make progress in fields such as High Energy Physics and Astronomy. In particular, we need to be able to:

- Learn how to access data stored in remote databases across the Internet.
- Learn how to filter the data to select the information that is of interest for our experiment.
- Learn how to organise the data so that we can apply relevant statistical tests.
- Learn how to present the processed data for human understanding.
- Learn how to use computational methods that can deal with large data volumes.

These *data analytics* skills are highly transferable, and are in wide demand (and command high salaries) outside the scientific world, particularly in finance.

4.2 Using Statistical Methods

Statistical methods are an essential tool in modern science, and important for anyone who wishes to work with Big Data. We often need to work with situations where, perhaps, there is an effect that might be worth knowing about, but it is masked by random variations (so how do we extract the underlying trend?), or we may see what appears to be a trend, but worry about the possibility that it might be an illusion—the result of a random clustering of data that just happens to look significant. (For example, bright

[6] Stations that include four detector plates see more events than those with just two plates—they just have twice as many chances to catch the edge of a shower. The Marling detector has just two scintillator plates.

stars are in truth randomly scattered across the sky, but from ancient times people have grouped them into the patterns we call constellations.)

There are several important statistical methods that we may need to use in order to analyse HiSPARC data. For the investigation of pressure (and possibly other) effects, however, we will rely principally on two approaches: *Regression* and *Analysis Of Variance* (also widely referred to as ANOVA).

Regression[7] helps us to fit the 'best' line to a graph of scattered points, and then tells us whether the line can be trusted to be a fair description of an underlying effect. You have probably done experiments in school where you have made measurements, plotted experimental points on a graph, then drawn a line through the points 'by eye'. Fortunately for you, most experiments in school laboratories are designed such that the effect you are investigating is very apparent, and there is a relatively small amount of measurement error. Most cutting-edge scientific research work, however, is done at the edge of measurability and measurement errors and random effects tend to be important. When we plot graphs, the points may be widely scattered. We may also have a great many points to plot (perhaps thousands). Nevertheless, we may still want to estimate the underlying correlation between two measurements (say cosmic ray event rate and barometric pressure)—but it may not be so obvious how to draw the best line through a blobby mass of points. We need a method of picking a line that in some precise mathematical sense is as close as possible to get to all the points on the graph.

There are good (but somewhat sophisticated) mathematical reasons for usually picking the line that minimises the sum of all the *squared* distances from the points on the graph to the chosen line. This is often referred to a 'least squares fitting', or 'least squares regression'. We often first try to fit the best straight line to the data, but if we suspect that the data show a more complicated pattern—say a quadratic—there are ways to find the best fit quadratic, or cubic, and so on.

Now that most experimental data is collected by computers, you never do least squares fitting manually, These days you never even need to write a program that does the job, because it is now always possible to find very good numerical library routines carefully constructed and tested by experts at numerical analysis. You can call up such library routines in Excel, Python, Java, Fortran, C, C++ and in fact just about any modern programming

[7] The name *Regression* came to be associated with the technique in a particular historical context. In now has no useful descriptive worth at all and it is not worth worrying what it originally meant. Just accept the word as one of those odd scientific names.

language. You only have to understand enough about regression to see how to provide these library routines with the right information, and to be able to interpret the results that are produced.

The library routine takes your arrays of X and Y values (better called the independent and dependent variables) and spits out the A and B coefficients of a straight line represented in the form Y = A.X + B, representing the underlying correlation. It also returns a *correlation coefficient*, usually called R, which tells you whether you have a good fit (or not). Decoding the full meaning of R would take us into A-level statistics and beyond. Suffice to say that R is a measure of the variation of the points away from the straight line, such that R=1.0 is a perfect fit, and R=0.0 is no useful fit at all.

I will go through the process that I would typically follow when looking at data of this type. I have picked up a few months worth of data from one of the Amsterdam Science Park stations (501). We do, however, need to repeat this process with more care using a much larger quantity of data, say, at least a year and preferably more.

I would first typically just plot the data against time to see if any patterns are obvious. So here is the event rate vs time. (See Figure 1 on page 16.) Two things stand out: firstly the average event rate is somewhere in the region of 2400/hr; secondly, there are big variations from day to day in the rate, and we might have a strong suspicion that these could be related to the weather.

So we might next plot the event rate against barometric pressure, and as part of this process I would try to fit a straight line (using the linear regression technique) to the scatter of points. Figure 2 shows a pretty strong correlation, and the straight line looks fairly good—though there may be a suggestion that the line should really have a slight curve, and we may want to revisit this fit later. Is this the whole story?

This leads us onto a widely used and extremely important statistical technique knows as Analysis of Variance (ANOVA), in which we try to explain all the scatter around our correlation lines by trying various correlations one by one until we get to a point where any remaining variations can be attributed to pure randomness that cannot be further explained.

Let me introduce a rule-of-thumb. If we believe that the rate of arrival of cosmic rays into the Solar System from the Galaxy is be completely random (and as far as we know, there is indeed no reason to expect them in correlated bursts) then in some hours we are likely to count more events than in other hours. There is no explanation other than the randomness. Some advanced mathematics (the Poisson Distribution—see below) now tells us that we can describe the way counts vary from hour to hour. That is, if we count events

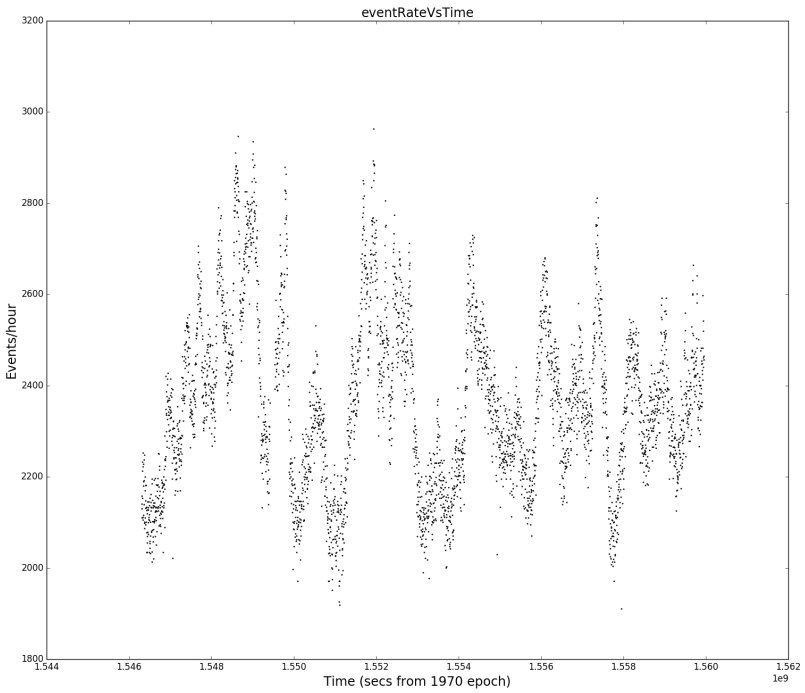

Figure 1: Cosmic Ray Rate Vs Time

in each hour (or in any other selected time interval) and get an average rate of, say, R, we can predict the chance of counting any particular number of events in future hours. (That is, over a very long future time, we could record for each hour all the counts and we would know what fraction of hours we would expect to yield a particular count value bigger or smaller than R.) *As a rule-of-thumb*, however, if we have an average event rate of R, we are very likely to see the actual count rate varying around R by $\pm\sqrt{R}$. (About 60% of the observed hourly counts would probably be within this range.)

Away from rules-of-thumb and back with proper statistically analysis, having plotting my set of points and obtained a regression line, my next step is a calculation of the *residuals*. That is to say, if my input data were represented by pairs of values (x_i, y_i) where, perhaps, the x represented

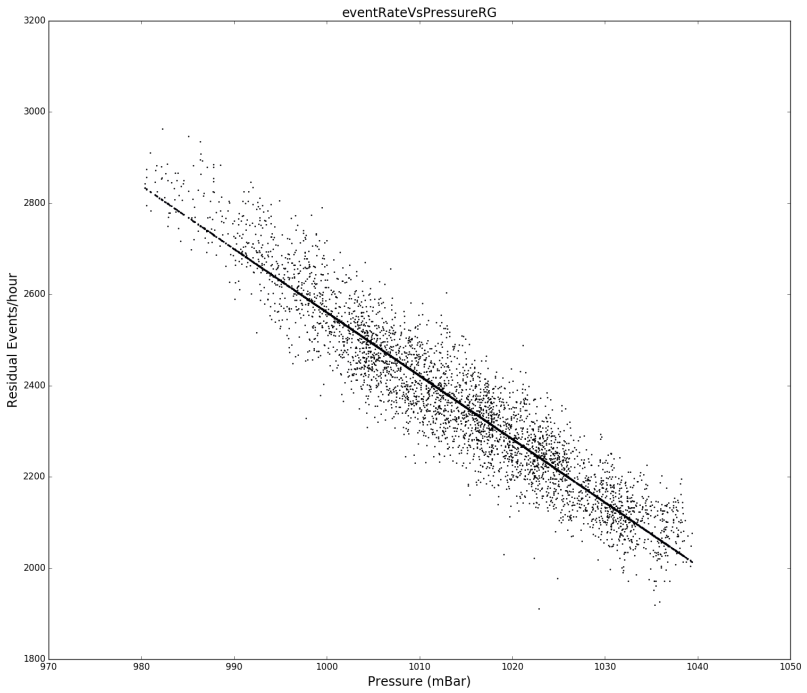

Figure 2: Cosmic Ray Rate Vs Barometric Pressure

pressure and y the count obtained over an hour at that pressure, and I obtained a fit $y = A.x + B$, my residuals, y_i^R, are calculated by subtracting the expected value of y, given the correlation, from the observed value.

$$y_i^R = y_i - (A.x_i + B). \tag{2}$$

This is the variation in the data *not* explained by the pressure correlation.

If what is left is pure randomness, when we plot the residuals against the original X (say, pressure) we should now obtain points scattered randomly around a horizontal line. If, it turns out, we ought to have been using a quadratic fit—a more complex relationship with pressure—then the distribution of points will not look straight. I am not going to include this graph here. Check it yourself.

I am actually now going to plot the residuals against time, after taking

account of the pressure correlations (see Figure 3 below). I can immediately see that there are still some patterns in the data, with variations from day to day and a trend over the whole period. Since this period covered later winter and early spring 2019, I might suspect that the average daily temperature was steadily rising and that we could also have a temperature effect on cosmic ray rate.

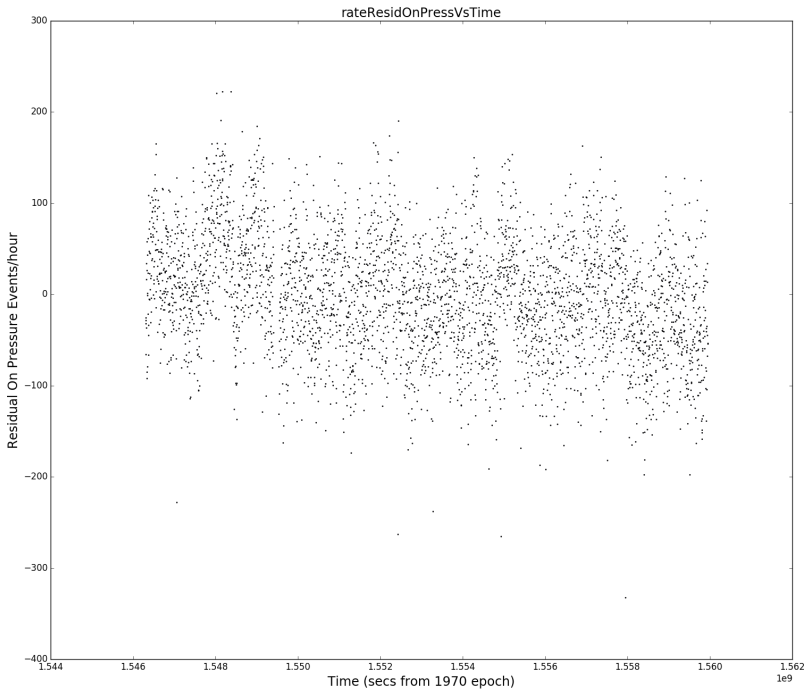

Figure 3: Cosmic Ray Rate Residuals (after Pressure-fit) Vs Time

At this point, therefore, I do another regression of these residuals against temperature. Indeed we do find that there is a correlation (not as strong as the pressure correlation, but still looking significant. At this point we do not know whether there is perhaps another correlation effect still to find. Is there anyway of eliminating this possibility without trying a long list of vague possibilities? (You need to be careful about trying lots of correlations where there is no prior reason to suspect dependency: sooner or later you

will find something in the data—and it may be just a random arrangement of points that fools you into making a claim that turns out to be wrong.) There is, however, a good way to see if there is information still buried in the data, even if we do not know what it is.

We can plot a histogram of these residuals to see if they have the type of distribution that we would expect from purely random arrival times. That is, we count the number of residuals that occur at various deviations from the known average count rate. Figure 4 shows what we have left after removing the correlations with pressure and temperature. Well, the residuals do indeed look as though they have a classic 'normal' distribution (the 'bell-shaped' curve—it is called 'normal' because it turns up so often in statistics). However, we need to consider whether the width of the distribution is consistent with what we might expect if there is nothing is left to explain other than the completely random arrival times of the cosmic rays.

If their arrival times are truly random, then it turns out that we can derive the probability that any particular period of recording (say one hour) will log a given number of events. We can, in fact, predict mathematically the exact shape of the histogram we would get if we plotted the residuals assuming that the cosmic ray arrival times were completely random. This is A-level/university level maths, so I am just going to quote a formula.

The French mathematician Poisson studied this problem back in the 19th Century and published a famous formula for what is now known as the *Poisson Distribution*:

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda} \tag{3}$$

where P(k) is the probability of getting k events in some time interval, where is the average number of events we expect in that interval, and e is the well-known Euler's number, 2.71828.... This formula is often useful when the average number of events in an interval is fairly low—say 10 or less. Although the reasoning behind this formula is usually considered university level mathematics, the formula itself is extremely useful to lots of people who would not know how to derive it.

If the average event rate is fairly high, then the formula still works, but looks more and more like a normal distribution. In fact, when the event rate is very large, say $\lambda > 1000$ (as it is for hourly cosmic ray counts) we

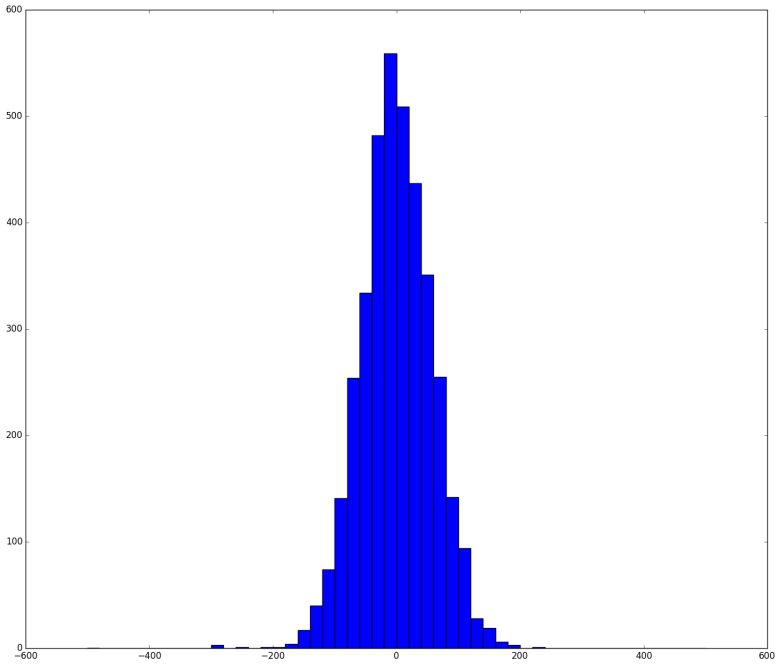

Figure 4: Histogram Cosmic Ray Rate Residuals (after Pressure and Temperature-fit)

can replace the above formula by one that is much easier to calculate[8]:

$$P(k) = \frac{1}{\sqrt{2\pi\lambda}} e^{-\frac{(k-\lambda)^2}{\lambda}} \tag{4}$$

This is indeed the same as the normal distribution that we talked about above, with a mean of and a standard deviation of $\sqrt{\lambda}$. (The standard

[8] Factorials, such as $k!$ and and powers, such as λ^k, are a problem to calculate directly on computers when k gets large (and not even particularly large—somewhere between 10 and 20) because their magnitude grows very quickly with k. and rapidly exceeds the ability of the computer to represent them accurately in their finite-precision arithmetic. There are, however, a number of useful mathematic tricks that help (such as working in logarithms) or using the well-known *Stirling Approximation*. These are discussed in any book on numerical methods, and also usually implemented in widely available library routines for calculating Poisson statistics, so you rarely actually have to do this yourself.

deviation is a measure of the width of a distribution such as the one we have illustrated in Figure 4.) As we noted above, our mean rate is about 2400, so we might expect the standard deviation of the curve to be a little less than 50.

As a matter of fact, I calculated the standard deviation for the data in Figure 4, and got a value of about 56—a bit larger than I might have expected for pure random data. There are a number of possible reasons why this can occur, including:

- We fitted straight lines to the correlations against pressure and temperature. They looked failry good, but it may be that we would have got an even better fit with, say a quadratic. (I suggested earlier that a slightly curved line might be a better fit to the data.)

- There might be an additional influence on cosmic ray event rate that we have not yet discovered—say, the intensity of the Solar Wind.

We must, however, beware of the danger of over-fitting. Correlations do sometimes occur by chance, and the more parameters we try to fit to our data the more likely it is that we will find something that looks significant. It is sometimes called the 'look elsewhere effect' because if you have not found something interesting with one set of fitting parameters, you keep trying different combinations and undoubtedly at some point you will eventually find an apparent but completely spurious correlation.

One the other hand, the standard deviation is not so very much higher than I would expect, and it may be that the particular selection of data I have used have given me a larger value by chance. At present I suspect that I do not have enough data to work out if anything else is going on. When I find that I am in this type of situation, and I know that there is more data potentially available, I go and get it, rather than worrying about applying complicated statistical tests. Things that are worth knowing about are usually obvious!

In summary, the ANOVA process involves taking out the successively most important influences from the observed data one by one until we are left with something that we are sure is pure randomness.

In professional research we usually try to estimate the probability that the results which we report could have occurred purely by chance. Standard statistical tests often return a so-called *P Value* which relates to the probability of getting the results we see by chance. For important claims, where being right or wrong matters more, we demand very small probabilities that our data could be due to chance. High Energy physicists (the people who

work at CERN for example) typically demand very low probabilities (say $P < 1/1,000,000$) before they will accept the 'discovery' of a new particle. Experiments on the Large Hadron Collider cannot be reproduced elsewhere at present—they are unique—and repeat runs required years of data taking on one of the World's most expensive machines. LHC Physicists need to really get it right first time. Biology and psychology journals, on the other hand, sometimes require P-value of only $P < 1/20$ before they let you publish a result because their experiments tend to be easier to check, so it is worth, in effect, saying in public 'We might have something interesting here. Please confirm it'.

You are however, not behaving scientifically if you keep repeating the experiments (say 100 times!) until one result looks significant[9]. Sooner of later something that looks positive will turn up just by chance.

The correct way to deal with this problem is to repeat the original experiment with a new set of data and see if the hypothetical effect is still there. In our case we can try the same experiment with a different HiSPARC station, or use data from a different time period on the same station.

4.3 Data Analysis with Programming

Writing error-free computer programs is challenging, and unless you are a well-trained professional, it is likely to take longer than you anticipate and unlikely to produce results that match your hopes. A large part of my job before retirement was ensuring that my colleagues in the nuclear industry did not stray into producing low quality programs when we needed very high quality software that worked well.

Therefore, whenever I had a problem that required handling of large amounts of data I would first look for professionally-developed tools (some of which, ah-hem, since I was, indeed a well-trained professional in the data handling game, I had previously produced myself). Nevertheless, it is not at all unusual to find that there is nothing available that quite meets your needs, because we continually wish to look at new types of correlation.

There is no way around it: if you are involved in experiments that produce large amounts of data, you need to know how to write computer programs. If, however, you are sensible you will also try to make the maximum use of proven tools produced by others.

[9] Unfortunately, this does happen. Experimenters often struggle to find which are the critical control parameters amongst many possibilities—and keep trying new variations until something pops-up). Other people then try to reproduce their findings and get nothing. In some journals about half of the published results turned out to be unreproducible!

I am likely to pose myself the following set of questions:

- Can we solve the problem using Microsoft Excel? I make no secret of the fact that I do not like using Excel for other than relatively trivial situations, mainly because they are often hard to test, and it can be very difficult to see if you have made mistakes when creating a complicated spreadsheet, Even when you are *sure* you have made a mistake somewhere, it can be almost impossible to find. Nevertheless, if I have a relatively small amount of data (no more than about 100 rows) organised in columns, and the manipulation required are relatively simple (multiply this column to that and add the other column) with the results then to be presented graphically as a simple x-y scatter plot, it can be the fastest and most reliable way to an answer.

- If the data is already stored in a relational data base to which I have access, can I do the data manipulations using the Standard Query Language (SQL)? Relational queries can sometime achieve with a one-line instruction what would require hundreds of lines of conventional procedural programming. It is well known that the time required to write programs and the chance of making mistakes rises linearly with the number of lines you have to write, so where it works, an SQL solution is often extremely efficient. (Sometimes it is even worth putting tabular data into a relational database just so we can manipulate it with SQL: I have done this many times.) Unfortunately, not all problems have a form that can be readily addressed using relational operations, and most scientists choose not to develop the skills to use relational languages. Well, HiSPARC data does get stored in a relational data base, but unfortunately we do not have direct access in a way that lets us use SQL queries.

- If I have a statistics problem, can I use one of the well-know and very reliable statistical toolkits? Statisticians and other 'data analytics' or 'data science' professionals may well use sophisticated (and expensive) commercial software such as SAS. Commercial organisations who pay such people high salaries usually conclude that it is also worth giving them the tools that can get them to answers as quickly as possible and so they are prepared to pay for the highly polished user interfaces, voluminous documentation and associated professional training courses. It works out cheaper in the end. For those who cannot afford this (such as many academic scientists) the Open Source 'R' statistical toolkit is just as (perhaps even more) powerful, but has a much less

polished user interface, and there is a lot of 'teach yourself' if you want to employ it.

- Can I find a software library that provides the tools that I need and that I can use with my favourite language? There are always compromises. Some computer languages and associated libraries, such as Python, are relatively easy to learn and have very comprehensive scientific and mathematical library support. That means you write less code which also means that it more likely to to be error free. On the other hand, because the design of languages such as Python emphasises rapid program building they cannot also provide efficient program execution—the programs may run relatively slowly, and that can matter if we want to process very large amounts of information. Other languages (e.g. C++) allow the construction of very efficient, fast running software but in my opinion are more complicated to learn and do not have the same range of scientific support. That means that although you may save time when executing a program, it may take you much longer to produce a program that works without errors.

  It comes down to a matter of judgement about which route will ultimately prove more efficient, given the amount of data you must process, the number of times you will want to do it in the future (I have produced software that had to be used everyday on every nuclear power station year after year) and the complexity of the problem and perhaps even your existing skills and the skills of the staff who are available to help you. (You might be a C++ expert and a Python novice. In which case go for C++.)

  Some languages, such as Javascript, though highly fashionable just at present (and even I quite like it for some things), are still relatively immature and have yet to acquire a comprehensive set of stable and well trusted numerical and statistical libraries. Fortran (almost the oldest of the computer languages) is still highly favoured in some areas of physics research because its syntax is well adapted to describing the type mathematical formulae that turn up in physics, it can be compiled for very efficient execution and also has very extensive numerical libraries that are tuned to work on very large scale parallel processing architectures. If you want to simulate the evolution of the universe, or the changes to the climate caused by global warming, this is probably the way you would go.

Only as a last resort do I program the data analysis algorithms myself. I

do, however, find that I frequently need to combine bespoke programs with analysis using one of the techniques above.

The bottom-line message: you need some basic knowledge of programming. The language you first choose to learn is less important because the higher-level programming skills are readily transferable between languages. As will all skills, the more you learn, the more you can do. The professional scientist working in large scale simulation or data analysis is probably fluent in several computer languages and switches easily according to which is the best for today's problem. He also knows how to write software systems where different parts are written in different languages, each using the language that is best adapted for the job it has to do.

4.4 Data Presentation

Good experiments and good analysis are of no use unless you can communicate the results effectively. That does not mean giving the reader every possible graph or table that you have produced. It does mean understanding the particular interests of the reader, and what he or she needs to know. It also means having an argument that you want to present, and knowing how to decorate the argument, expressed in words, with the right figures at the right places. You can hide the point that you really want to get across if you bury it in piles of irrelevant graphs.

As always, the standard rules of producing graphs must be followed:

- Each axis must be labelled and show units.
- The graph should have a title.
- If you are plotting several curves on the same graph, provide a legend to explain which is which.
- Remember when plotting multiple curves on the same graph that a not insignificant proportion of readers are colour blind (there is probably at least one in your class—who may not even realise it) so you should not use only colour to distinguish the curves. It is a good idea to also employ different types of broken line (dot-dash patterns) or different icons for points.
- Every figure in a document should have a figure number, and should also be referenced from and explained in the text.

Once again, if you are a professional with a serious research budget you might well purchase a slick commercial graphics package. Many academics

prefer to spend their money on other things, and use open source software which is less easy to use, but often just as powerful and flexible. There are many such packages available on the Web, perhaps hundreds, and you may already be familiar with one of these tools, or have one already installed on your computer. I can only speak about those that I have used extensively, and that I know work very well. They are also ones that are very widely used and almost always come somewhere in the ubiquitous lists of 'top-five free graph plotting packages'.

It is not enough, however, just to plot a graph. We also need to be able to incorporate it into our reports. That may well influence the tool we choose because some produce graphs that are easy to include in Word documents, while others are better at interfacing with the other document production tools that are more widely employed by physicists and astronomers.

4.4.1 GNUPlot

GNUPlot (gnuplot.org) is extremely widely used in research establishments. It can probably do anything most researchers need (outside the realms of producing 3D reconstructions of brain-scans and so on). It is relatively easy to learn to do simple plots, and although you have to teach yourself how to use it from the manual (and a few YouTube videos) the manual is well written and very comprehensive. Your main problem is finding the description of the option you need to employ, amongst many other options.

It is particularly liked by mathematicians and physicists who tend to write a lot of documents using a system called LaTeX, because it produces its graphs in a form that are easy to incorporate into LaTeX documents. When used in this way it is designed to produce 'publication quality' graphs (i.e. acceptable to editors of scientific journals who need to have figures that reproduce well in print without taking up too much space).

If you have computer files containing tabular data in columns then GNUPlot is probably the simplest way to get to an elegant graph. It is available, free, for all types of computer systems and is likely to be pre-installed with many versions of Linux (though installing on MacOS in not entirely trivial, as it is with Windows).

4.4.2 matplotlib

Those who program in Python can use one of several library 'modules' for graph plotting. I have used *matplotlib*. It is, in my opinion, not as easy as GNUPlot, unless you requirements mean that you can just directly copy one

of the recipes from the on-line tutorials. That will, however, probably get you through about 90% of what you will ever need to do. The difficulties occur if you want to go to the next stage and do something not explained in the tutorials. Then you will need to spend a good deal of time learning how you build graphs from simpler components[10].

Nevertheless, *matplotlib* is very useful if you are do a lot of data processing with Python because you do not have to write out files containing data tables just so you can transfer data to applications such as GNUPlot. It is usually much easier just to pass the arrays holding your data to the relevant *matplotlib* function. You can, if you need, produce very sophisticated layouts, fully up to the standard required by editors of scientific journals and it is highly configurable so that you can take account of the preferences of different journals.

The graphs in this note were produced by *matplotlib*, but they also illustrates one of the problems with this package—the default text for axis labels can come out too small, and experimentation with the configuration is usually required for the best results.

4.4.3 Excel

Microsoft Excel is very convenient for producing plots from relatively small datasets. As long as you are prepared to accept one of the standard formats it is very easy to use, but configuring it to do something slightly out of the ordinary can be quite challenging.

Excel is also very convenient if you want to embed graphs in documents produced with other Microsoft software, such as Word. It is *much* less convenient if you are a mathematician or mathematical physicist who needs to write document full of equations (i.e. probably using LaTeX rather than Word). It is not so easy to persuade Excel to save graphs in one of the formats used by almost everyone other than Microsoft. We do it if we have to, but we do not enjoy it.

One other big disadvantage for scientists handling large amounts of data is that Excel is very slow if you have more than a few hundred lines, and graphs based on tables with more than a few hundred rows tend to come out as overlapping clouds of points and are completely useless for scientific reports. I have yet to find a way to persuade Excel's graph plotter to use

[10] My opinion of *matplotlib* may be partly because I never had cause to learn to use a commercial software package called MATLAB that is widely employed by scientists and engineers, because *matplotlib* is designed to mimic features of MATLAB. This is very convenient is you are an experienced MATLAB user, but less so if you are not.

really tiny points.

A Deriving EAS/Steradian from Detector Signals

The detectors are horizontal plates, each of about 0.5 square meters. It is, however, only showers arriving from the zenith that see the full area, at any other zenith angle, θ the projected area on the sky is reduced by $cos(\theta)$. That will reduce the EAS trigger rate by $cos(\theta)$ so we will need to divide by a factor of $cos(\theta)$ to infer the trigger rate from a plate placed perpendicular to tracks coming in at zenith angle θ.

In addition, however, there is more sky at larger zenith angles than smaller angles, just because we are not distinguishing between EAS events at different azimuthal angles. So, between zenith angles θ and $\theta + d\theta$ on a sphere or radius R there is an area $2\pi Rsin(\theta)Rd\theta$. This represents a solid angle of $Area/R^2 = 2\pi sin(\theta)d\theta$, and for a detector that is uniformly sensitive to all directions this would describe the variation in detection rate for EAS events with zenith angle.

Our detector, however, is not uniformly sensitive, so we should include the $cos(\theta)$ factor for the projection of the detector plate area to get a factor of $2\pi sin(\theta)cos(\theta)d\theta$ for the variation in expected EAS event rate with zenith angle for an infintessimal range of angles between θ and $\theta + d\theta$.

In practice, we want to divide our detection rate into finite ranges of zenith angle—say five or ten degrees. We therefore need to sum the areas using an integral between the upper and lower angle:

$$F = \int_{\theta_1}^{\theta_2} 2\pi sin(\theta)cos(\theta).d\theta \tag{5}$$

For those who have done A-level maths (integral calculus) this is an integral that can be immediately reduced to a trivial form by a well-known substitution trick, $cos(\theta)d\theta) \rightarrow d(sin(\theta))$ giving

$$\int_{\theta_1}^{\theta_2} 2\pi sin(\theta)cos(\theta).d\theta = 2\pi \int_{\theta_1}^{\theta_2} sin(\theta).d(sin(\theta)). \tag{6}$$

which is now, after the substitution $sin\theta \rightarrow x$, has the trivial form $\int x.dx \rightarrow x^2/2$. For those who have not yet done A-level maths, you will learn to do stuff like this in your sleep.

Hence:

$$\int_{\theta_1}^{\theta_2} 2\pi sin(\theta)cos(\theta).d\theta = \pi\left[sin^2(\theta_2) - sin^2(\theta_1)\right] \tag{7}$$

This clearly has the right sort of form, because it goes to zero at the zenith (where there is an infinitesimal amount of sky) and also goes to zero at the horizon (where there is zero projected detector plate area) and is a maximum half-way between.