# Setting up a data taking procedure

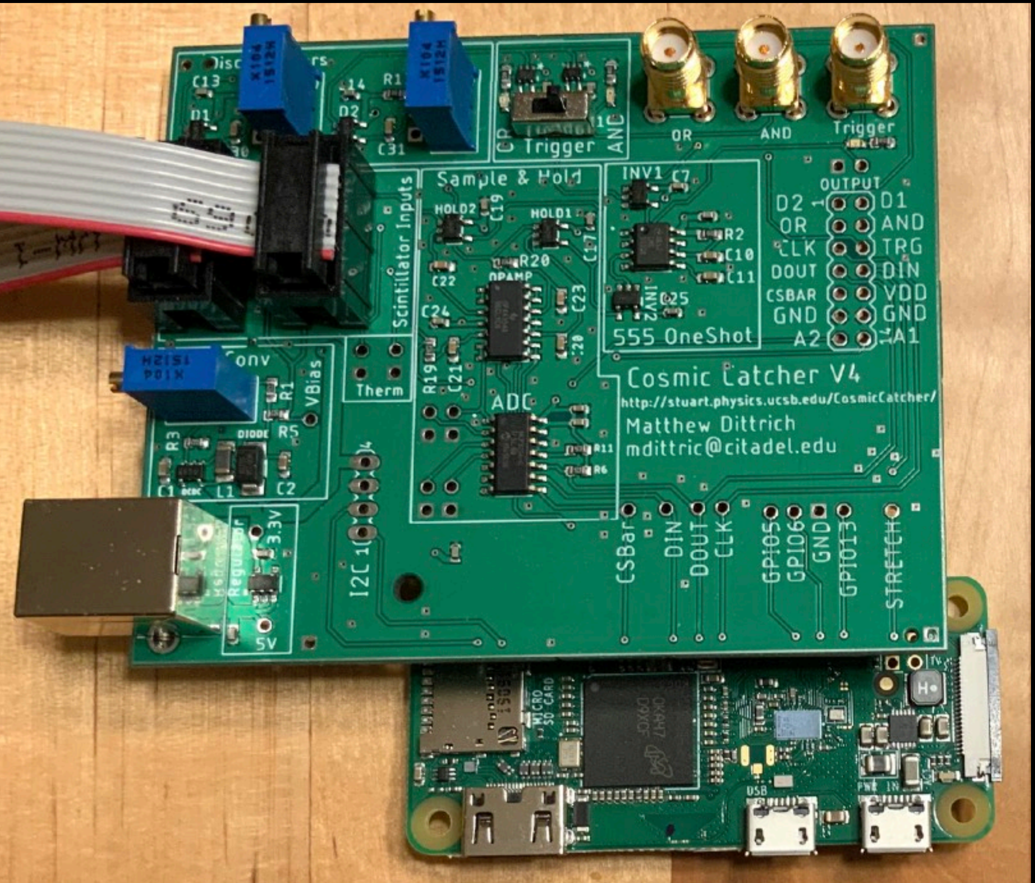
Phys150 Special topics

David Stuart, UC Santa Barbara

## Review: Detector board

We went through the function of the board to understand it.
Final exam may have some simple questions to check understanding.
Think of it like an interview question: did you understand your work?

Saw that simple things could be learned quickly from LEDs, but needed to collect data to get more quantitative results.

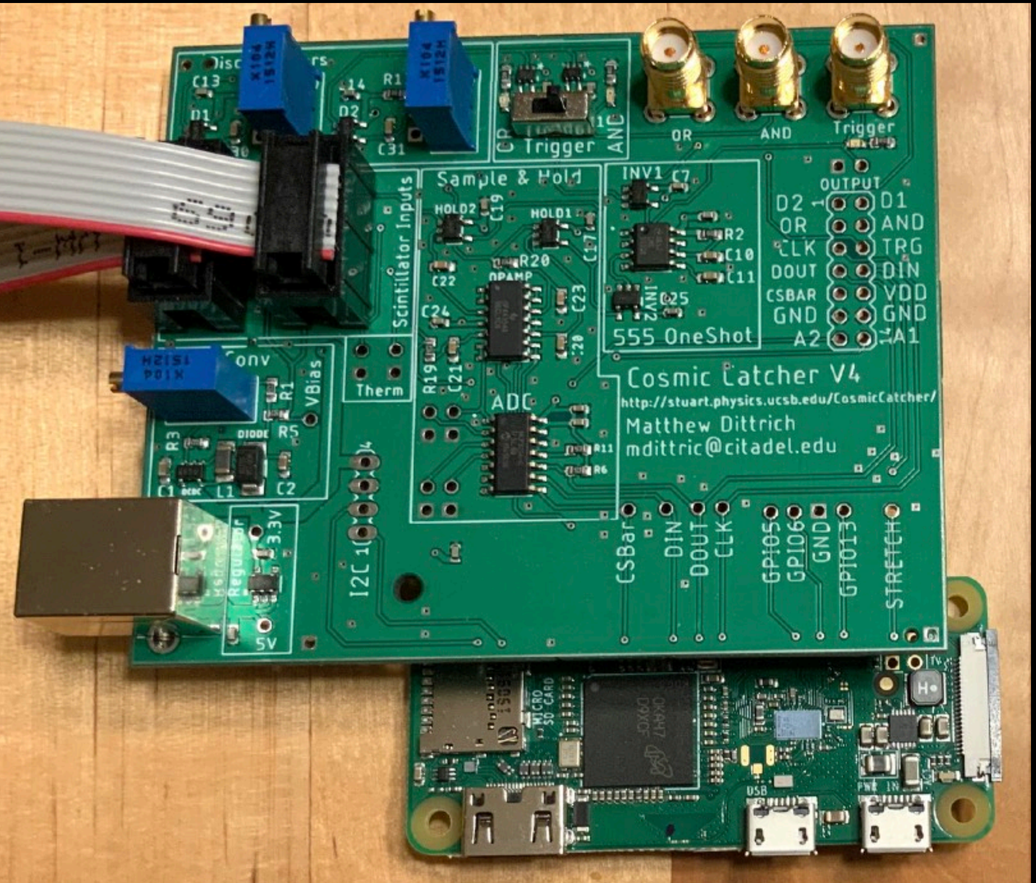


Specifically we saw:
- Light leaks
- Uncertainty on rate from sqrt(N)

## Review: Detector board

We went through the function of the board to understand it.
Final exam may have some simple questions to check understanding.
Think of it like an interview question: did you understand your work?

Saw that simple things could be learned quickly from LEDs, but needed to collect data to get more quantitative results.



Specifically we saw:
- Light leaks
- Uncertainty on rate from sqrt(N)

The sqrt(N) is a *statistical* uncertainty, ie random fluctuations.

There are also *systematic* uncertainties, ie unknown biases.
E.g., light leaks, stable or varying.

## Developing a more rigorous data taking plan

We can’t just make one measurement and be done.

Suppose you wanted to test whether there is a day-night difference.
It would be simple to tell if it completely goes away, just with LEDs.
But could you confidently observe a 10% difference? 1% difference?

  Could be fooled by varying ambient light getting into leaks.
  Maybe the bias voltage or threshold voltages drift a bit.
  Maybe there are temperature dependencies.

A careful measurement requires controlling for all these things,
by measuring them.

So we want to be able to take a large number of individual measurements and use them to extract a conclusion. Typically this means measure rate as a function of X, R(X), to determine its dependence, but also measure and control for other dependencies such as R(Y), R(Z), R(t), R(T), …

## Developing a more rigorous data taking plan

That means we need to *repeatedly* measure the rate, keep track of the results, and the run conditions, and then analyze them globally.

We should also make sure that the rate does not change with time more than expected from statistical uncertainties.

Measure it 100 times and see if it varies as expected within uncertainty. And we should probably do this regularly.

This motivates the idea of a “run” which is a set of data taken with a specific configuration.
You may change configuration between runs, or maybe not, and then compare the rate measured in different runs.

## Runs

As a test of that, let’s take 60 runs with 60 seconds of counting per run. We can calculate an average rate for each run and check whether it is constant with run number, ie time.

But, this could be cumbersome to collect a lot of separate runs and keep track of everything.

So, automate it.

## Scripts

Let’s write a program that runs our data taking program 60 times.

This is easy to do in linux; indeed that is built into its design.
It is a set of tools and a command line interface that is an interpretive programming language. That is called the “shell”.

There are several options: `sh, bash, csh, tcsh, zsh`.
The default is bash, I prefer tcsh because it is based on the C language.
You can use any one you want, but I’ll use tcsh for uniformity now.
We can change the default with

```
  sudo nano /etc/passwd
```

Then we can use it to write little programs to automate things.

## Example script

A first example:

```
@ N = 1
while ($N <= 20)
 @ TwoN = $N * 2
 echo “$N times 2 = $TwoN”
 @ N = $N + 1
end
```

## Example script

A second example:

```
@ N = 30
while ($N < 50)
 ping -c 1 -W 1 128.111.19.$N > /dev/null
 if ($status == 0) then
  echo -n "IP address 128.111.19.$N is "
  host 128.111.19.$N | awk '{print $5}'
 endif
 @ N = $N + 1
end
```

## Script to take data

So we can take 60 runs with something like

```
@ N = 1
while ($N <= 60)
 ./TakeData.py
 @ N = $N + 1
end
```

But we need to write the TakeData.py program in a way that it saves the new data rather than overwriting the old data.

## Script to take data

So we can take 60 runs with something like

```
@ N = 1
while ($N <= 60)
 ./TakeData.py
 @ N = $N + 1
end
```

But we need to write the TakeData.py program in a way that it saves the new data rather than overwriting the old data.

I will make it write to a single output file summary.dat that I can then rename to match the run number…

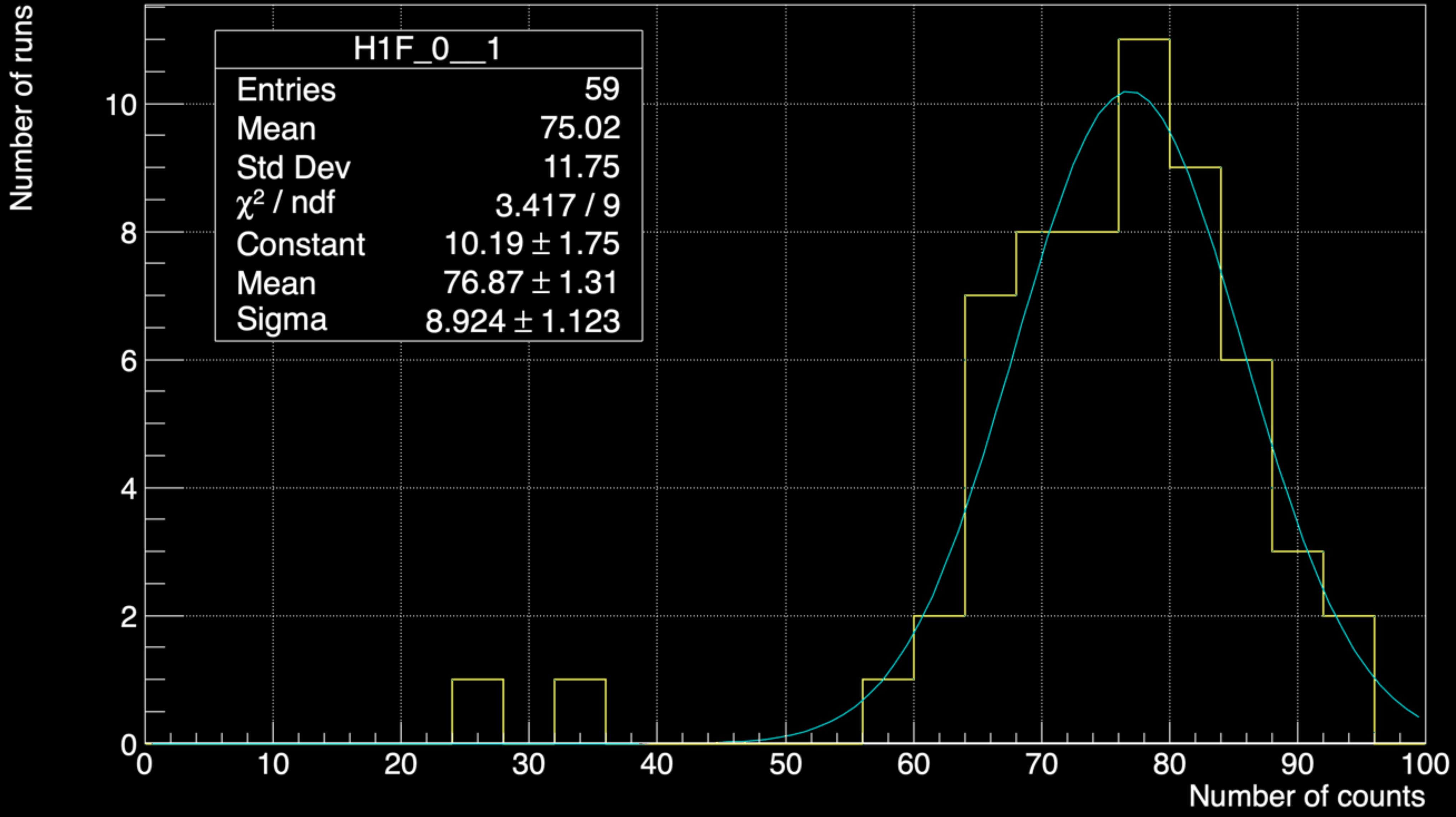
## Script to take data

So we can take 60 runs with something like

```
@ N = 1
while ($N <= 60)
 ./TakeData.py
 mv summary.dat Run${N}.summary
 @ N = $N + 1
end
```

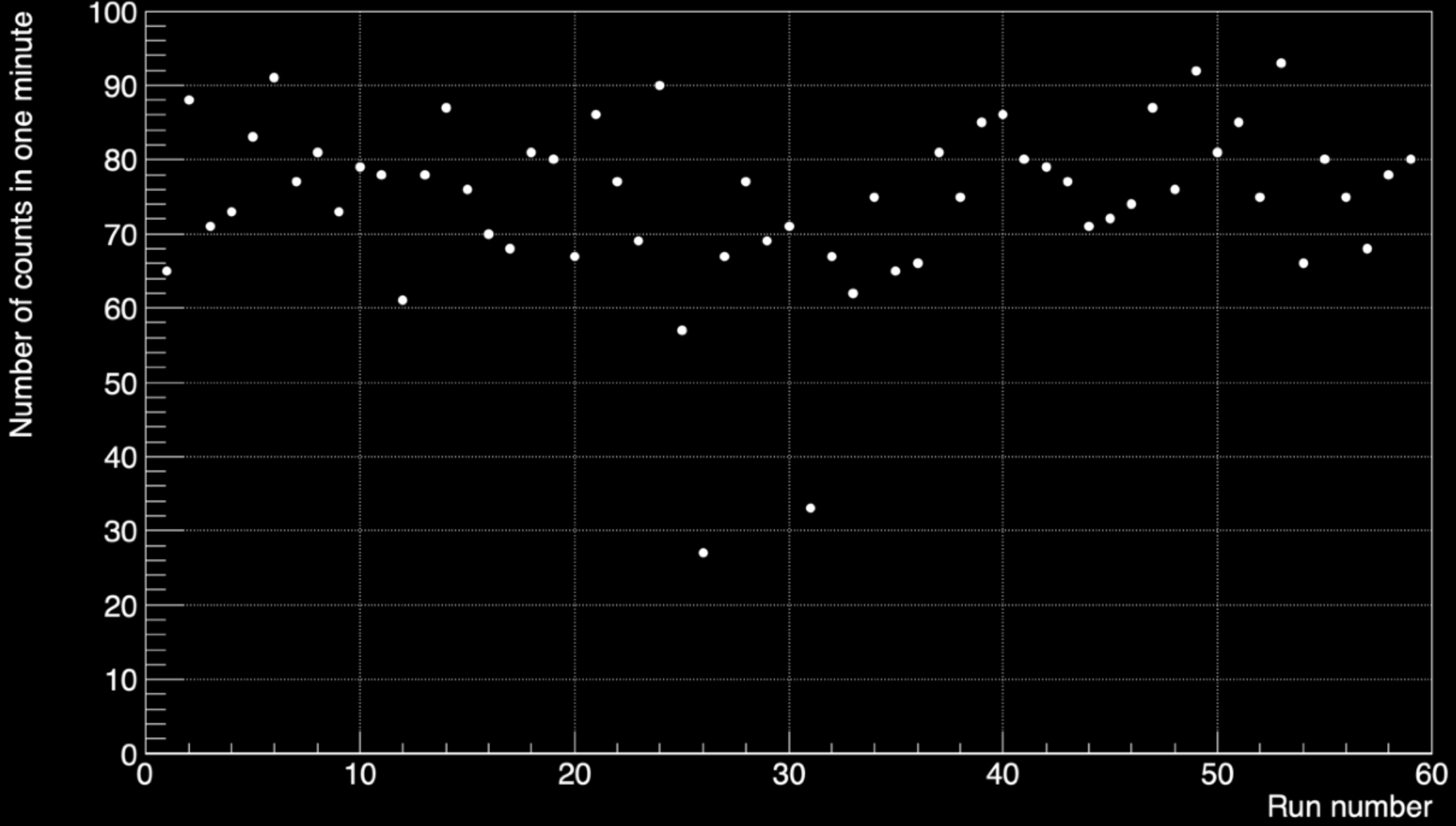
So, let's look at this program…

## Script to take data



## Script to take data

Outliers likely due to something anomalous going on during that time.
They are clearly anomalous and should be removed from a calculation of average rate during this period.
But it is worth noting them, with a likely hypothesis for the problem, and possible follow up later.
I happened to be using the RPi for other things then and probably bogged it down.



## Script to take data

Since we are going to want to take a lot of runs in different configurations, it is worth developing an automated system to handle all of the accounting.

Want it to be:

- easy to execute a measurement
- full record of configuration
- full record of data
- easy to look up what we did later
- easy to compare runs
- automated

We'd like an interface that is something like issuing a command such as

```
DoRun 10 "Bars oriented vertically. Lights off. Vbias=28."
```

Where "10" is the time to run and the rest is a comment for us to remember later. Then the RPi starts a new run, records configuration, and puts the results on the web.

And, we'd like a log of everything that it does, or that we do.

## Data taking software suite

We can build a system to do this by connecting a suite of tools, just like unix does. So, what are the actions we will take?

```
DoRun
GetNewRunNumber
LogConfig
Comment
StopRun
PublishRun
```

Then later you might want to add analysis and monitoring tools like:

```
PlotRun        GetRunConfig       CompareRuns
```

We may also want to add more sophisticated things **later** such as:

```
Online monitoring
Synchronize clocks with collaborating RPis.
Expanded data structure
```

## Data taking software suite

This requires a bit of structure. We need the following:

A standard place to put the data and file naming convention.
A place to save configuration information, ie a database.
A place to save the scripts that implement the actions.
An agreed upon data format.

Let's agree on a data format:
  Full information for each detected event, ie an ntuple.

```
    Event# FullTimeInfo
```

  Summary information for each run, ie another ntuple.

```
    StartTime EndTime RunTime NumEvents Rate RateErr
```

We might want to expand the format later to include new, auxiliary information, e.g., pulse height information from the ADC, altitude, etc.
So, let's leave space for expansion now with (initially empty) elements.

## Data taking software suite

This requires a bit of structure. We need the following:

A standard place to put the data and file naming convention.
A place to save configuration information, ie a database.
A place to save the scripts that implement the actions.
An agreed upon data format.

Let's agree on a data format:
Full information for each detected event, ie an ntuple.

```
    Event# FullTimeInfo ADC1 ADC2 AUX1 AUX2 AUX3
```

Summary information for each run, ie another ntuple.

```
    StartTime EndTime RunTime NumEvents Rate RateErr
```

We might want to expand the format later to include new, auxiliary information, e.g., pulse height information from the ADC, altitude, etc.
So, let's leave space for expansion now with (initially empty) elements.

## Where to put the data

Let's put all the programs in `/home/pi/150/bin`
Let's put the data in `/home/pi/150/data`
Let's put the database information in `/home/pi/150/db`
Let's put the logging information in `/home/pi/150/log`

The database directory will contain files keeping track of:
- the next run number,
- current default configurations (eg Vbias, Vthr1, Vthr2)

The log directory will contain a time-stamped & run-stamped log of actions taken by programs or by you.  Let's write a tool called `Log` that adds a line with whatever information we want. Eg.

```
Log "Changed Vbias to 29 V"
Log "Turned off the room lights"
```

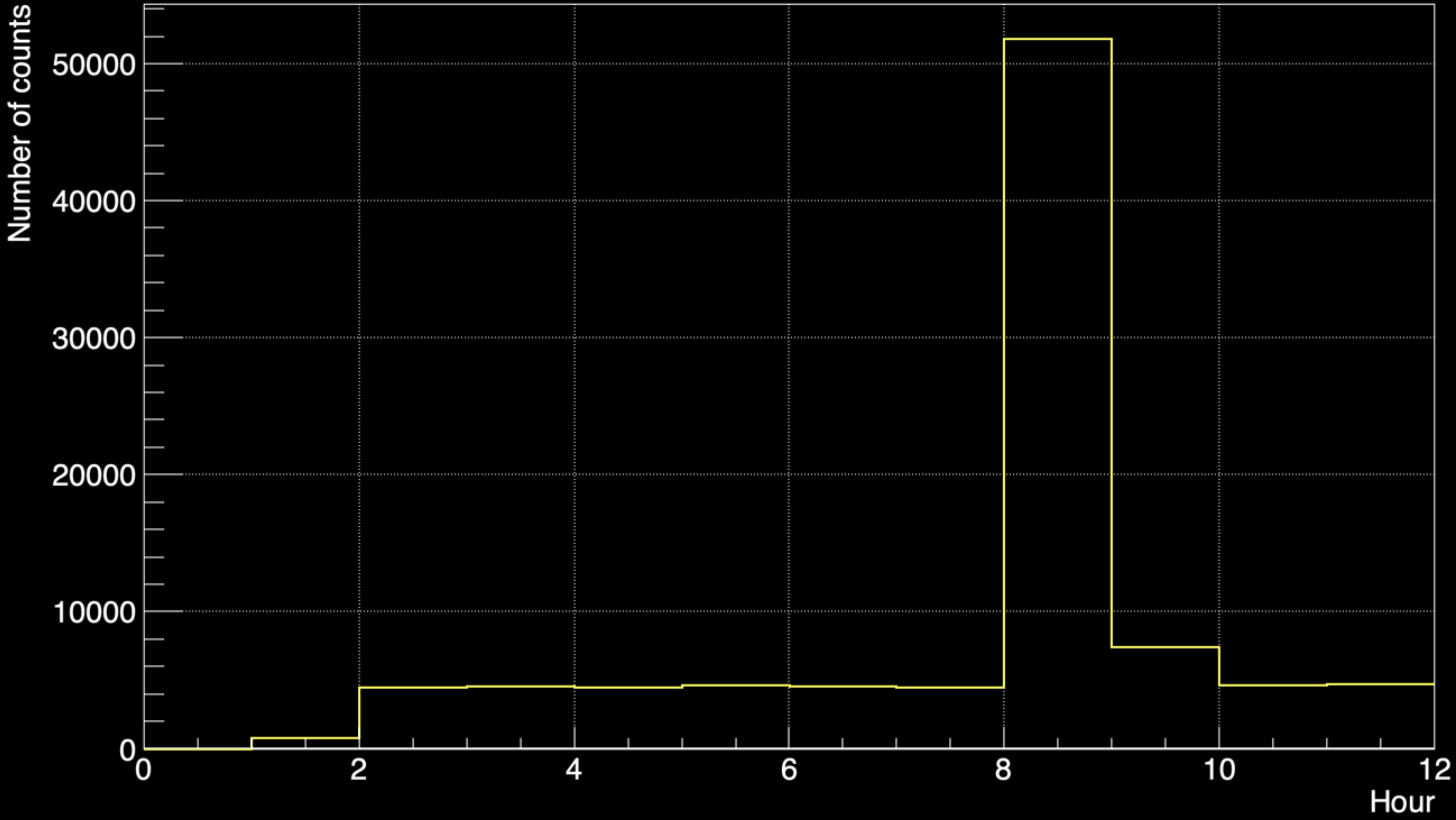
The data directory will contain a separate directory for each new run.

## Where to put the data

Demo creation of these directories.
Write a shell script for `NewRunNumber` and `Log` and `DoRun` tools.

## An overnight run

I took an overnight run. Here is the distribution of hour for all counts, ie hit rate vs time.  What do you conclude from this?



## Homework

This week's homework assignment is to write a summary of what cosmic ray property you want to measure and a sketch of how you will do it.

Post this description in your ELog before Tuesday's class.

For lab this week, set up your data taking tools and take an overnight run. Analyze the time stability thinking about statistical and systematic uncertainties.  Due next Friday.