Data Preparation

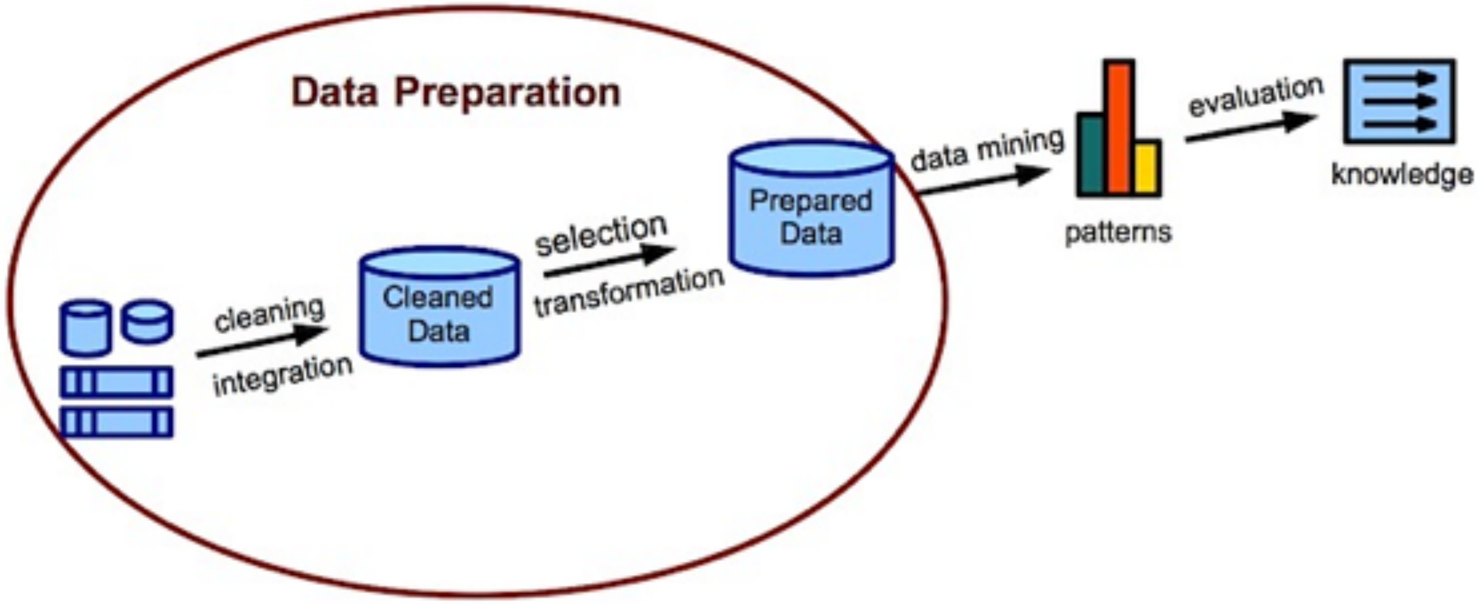
Riccardo Di Sipio, University of Bologna and INFN

Outline

- What is data preparation? Why do we need it?
- formatting, transformation, reduction
- Real-life examples
- Exercise

What is data preparation?

- Data need to be manipulated prior to analysis
- Poor quality data typically result in incorrect and unreliable analysis results



garbage in, garbage out!

Manipulation is…

- to expose data in a suitable format
- to select only useful entries
- to remove “bad” entries
    - bad = malformed, taken under faulty conditions, outliers, etc..

Data formatting

Plain text format

- Pros:
  - Typically human-readable on small-scale files
  - Easy to create/modify
- Cons:
  - Hardly scalable, highly non-standard
  - Write parser from scratch, always
  - Maybe not best option for machines:
    - large data files
    - unclear how to store non-trivial data

Advanced text formats

- CSV (comma-separated-values)
- XML (extensible markup language)

CSV

- fields separated by a comma or other delimiter

```
id,name,email
0,riccardo,disipio@cern.ch
1,mario,balotelli@cern.ch
```

- human readable
- parsers available for most languages
  - custom parser “from scratch” easy to implement
- recognized by many commercial programs (excel)
- unclear how to store non-trivial data

XML

- Markup language, generalizes HTML

```
<?xml version="1.0" encoding="UTF-8" ?>
<usersdb>
    <user id="1">
        <name>Riccardo</name>
        <surname>Di Sipio</surname>
        <email type="work">disipio@cern.ch</email>
    </user>
</usersdb>
```

- Highly scalable, simple, general
- Defines standard for document formatting
- Human- and machine-readable
- Parsers available for most popular languages
- Lots of typing if writing from scratch - bad idea anyway

CSV vs XML

- Example: define ROOT histograms

C++ code

```
TH1F * h_jet_n    = new TH1F( “jet_n”, "No. of Jets”, 15, -0.5, 14.5 );
TH1F * h_jet_pt   = new TH1F( “jet_pt","Jet p_{T}”, 20, 0., 1000.);
Double_t edges[5] = { 300., 500., 800., 1100., 1500. };
TH1F * h_fjet_pt  = new TH1F( “fjet_pt”, "Fat Jet p_{T}”, 4,  edges );
```

CSV

```
name,title,nbins,xmin,xmax
jet_n,No. of Jets,15,-0.5,14.5
jet_pt,Jet p_{T},20,0.,1000.
fjet_pt,Fat Jet p_{T},4,300.:500.:800.:1100.:1500.
```

XML

```
<histograms>
  <TH1F name="jet_n" title="No. of Jets" nbins="15"  xmin="-0.5" xmax="14.5" />
  <TH1F name="jet_pt" title="Jet p_{T}"  nbins="20" xmin="0."   xmax="1000." />
  <TH1F name="fjet_pt" title="Fat Jet p_{T}"  nbins=“4" edges="300.,500.,800.,1100.,1500." />
</histograms>
```

ROOT Trees (HEP)

- Structured files, contain one or more data containers (tree). Supports compression, distributed files system (XRootD)
- Trees are “tables” containing a series of entries (e.g. events) (rows)
- Each entry has a number of fields (e.g. pT, eta, phi, E, m, q) (columns)
- Not only numbers! Class instances, e.g. histograms or other complex objects
- See also Ivo’s talk

```
disipio: root ntuple.root
root [0]
Attaching file ntuple.root as _file0...
root [1] .ls
TFile**     ntuple.root
 TFile*     ntuple.root
  KEY: TTree    data;1  Arduino sensors data
root [2] data->Print()
******************************************************************************
*Tree    :data      : Arduino sensors data                                   *
*Entries :       25 : Total =            2994 bytes  File  Size =       1138 *
*        :          : Tree compression factor =   1.42                       *
******************************************************************************
*Br    0 :id        : id/I                                                   *
*Entries :       25 : Total  Size=        631 bytes  File Size  =        130 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.30     *
*............................................................................*
*Br    1 :timestamp : timestamp/I                                            *
*Entries :       25 : Total  Size=        666 bytes  File Size  =        141 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.25     *
*............................................................................*
*Br    2 :volt      : volt/F                                                 *
*Entries :       25 : Total  Size=        641 bytes  File Size  =         95 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.80     *
*............................................................................*
*Br    3 :temperature : temperature/F                                        *
*Entries :       25 : Total  Size=        676 bytes  File Size  =        122 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.46     *
*............................................................................*
```

Missing Fields

- Input data organized in records (entries)
- Some fields may have missing values
- How to treat these cases? (value = ?)

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	high	?	no	yes
2	very_high	yes	yes	yes
3	?	no	no	no
4	high	yes	yes	yes
5	high	?	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	?	yes	?	yes

Missing Values

- A missing value may be present because:
  - it is not available at the data taking (device off-line, person refused to answer)
  - it was mistakingly erased
  - Noisy communication channel
- Action need to be taken:
  - Discard the entire dataset
  - Remove/skip entries containing missing fields
  - Assign default value (NULL,0,1,out-of-range)
  - Extrapolate from the other non-empty fields
  - Statistical evaluation, random

Missing Values

- Delete entries

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	high	?	no	yes
2	very_high	yes	yes	yes
3	?	no	no	no
4	high	yes	yes	yes
5	high	?	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	?	yes	?	yes

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	very_high	yes	yes	yes
2	high	yes	yes	yes
3	normal	yes	no	no
4	normal	no	yes	no

Missing Values

- Replace missing field with most common value

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	high	?	no	yes
2	very_high	yes	yes	yes
3	?	no	no	no
4	high	yes	yes	yes
5	high	?	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	?	yes	?	yes

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	high	yes	no	yes
2	very_high	yes	yes	yes
3	high	no	no	no
4	high	yes	yes	yes
5	high	yes	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	high	yes	yes	yes

Missing Values

- Replace missing field with mean value

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	100.2	?	no	yes
2	102.6	yes	yes	yes
3	?	no	no	no
4	99.6	yes	yes	yes
5	99.8	?	yes	no
6	96.4	yes	no	no
7	96.6	no	yes	no
8	?	yes	?	yes

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	100.2	yes	no	yes
2	102.6	yes	yes	yes
3	99.2	no	no	no
4	99.6	yes	yes	yes
5	99.8	yes	yes	no
6	96.4	yes	no	no
7	96.6	no	yes	no
8	99.2	yes	yes	yes

need numerical values!

Missing Values

- Replace missing field with all possible values

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	high	?	no	yes
2	very_high	yes	yes	yes
3	?	no	no	no
4	high	yes	yes	yes
5	high	?	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8	?	yes	?	yes

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1^{i}	high	yes	no	yes
1^{ii}	high	no	no	yes
2	very_high	yes	yes	yes
3^{i}	high	no	no	no
3^{ii}	very_high	no	no	no
3^{iii}	normal	no	no	no
4	high	yes	yes	yes
5^{i}	high	yes	yes	no
5^{ii}	high	no	yes	no
6	normal	yes	no	no
7	normal	no	yes	no
8^{i}	high	yes	yes	yes
8^{ii}	high	yes	no	yes
8^{iii}	very_high	yes	yes	yes
8^{iv}	very_high	yes	no	yes
8^{v}	normal	yes	yes	yes
8^{vi}	normal	yes	no	yes

Resulting table may be inconsistent!

Specify a set of rule to deal with such cases

Data transformation

Why?

- Communication bandwidth is always limited → compress data
  - *bit streams, binary formats. need unpacking*

- Expose data into a different, standardized representation → transform data
  - *normalization, structure of arrays (SoA), arrays of structures (AoS)*

- Structure of Arrays (SoA) or Array of Structures (AoS)?
- See also this page

```
typedef struct {
   double pT;
   double eta;
   double phi;
   double E;
   int    q;
} Particle;
Particle particles[3];
```

```
typedef struct {
   double pT[3];
   double eta[3];
   double phi[3];
   double E[3];
   int    q[3];
} DataList;
DataList data;
```

Preferred by humans
Objects “have” properties

Preferred by computers
Improve memory utilization
Gain from vectorization/SIMD

double	double	double	double	int
double	double	double	double	int
double	double	double	double	int

double	double	double
double	double	double
double	double	double
double	double	double
int	int	int

Why?

- Communication bandwidth is always limited → compress data
  - lossless compressions:
    - data can be recovered 1-1
    - error rate should not increase before/after
  - information loss tolerable: jpeg, mp3
- Overhead < size(data)
- Redundancy (*e.g.* noisy channels)

A lossless compression gone bad

Original data encapsulated as ROOT TLorentzVector - large overhead, flexible

Data transformed in plain ntuple - just a list of float/int values

TLorentzVector → (px,py,pz,E)

TLorentzVector → (pT,eta,phi,E)

which repr. do you chose?

overhead
redundancy
flexibility

- Option #1: sum of momenta straightforward (*e.g.* reconstruct inv. mass)
- Option #2: most common representation. quick to fill histograms

Option #2 was chosen to spot potential errors at glance (weird pT values).
Unfortunately, analyzers had to convert the four-momenta to option #1 all the times because of interfaces to external libraries
Redundancy was not an option due to limited disk space

discuss first what final users want!

When?

- Device $\Rightarrow$ Computing devices (bit stream)
- Peer-to-peer transmission over network
- Disk space is an issue

Where?

- On-board, firmware
- Servers

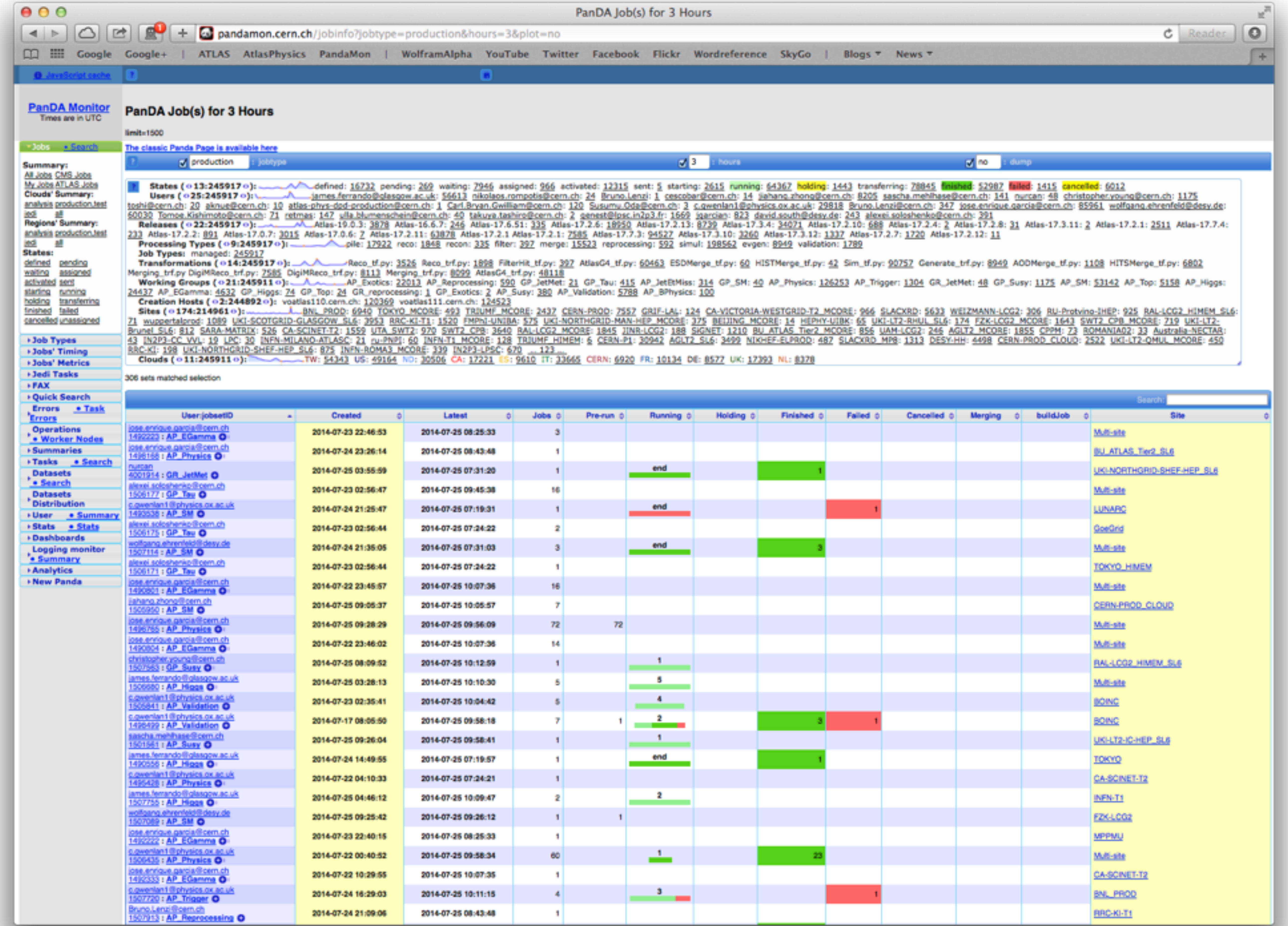
Data reduction

Why?

- Not all data are interesting!
  - Good run list: discard events taken under faulty conditions (busy/offline/tripped sub-detectors)
  - Reconstruct $Z \rightarrow \mu\mu$. Acquire events with a single-lepton (μ) trigger, discard those with only 1 μ

Where

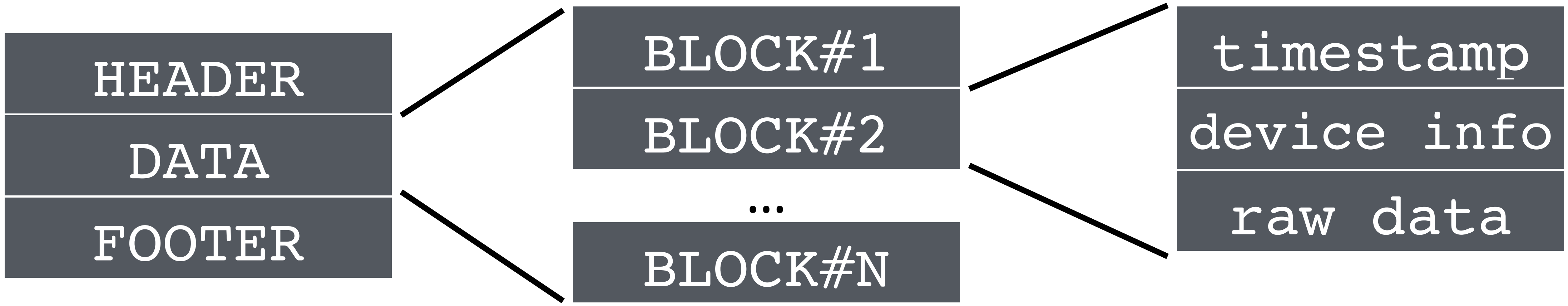
- Computing farm, CERN Grid
- “Distill” only interesting events, then store them somewhere else typically closer to the final user
- Bookkeeping very important for large datasets!


PanDA Job(s) for 3 Hours
pandamon.cern.ch/jobinfo?jobtype=production&hours=3&plot=no
Google Google+ ATLAS AtlasPhysics PandaMon WolframAlpha YouTube Twitter Facebook Flickr Wordreference SkyGo Blogs News
PanDA Monitor
Times are in UTC
PanDA Job(s) for 3 Hours
limit=1500
The classic Panda Page is available here
production
306 sets matched selection
User:jobsetID
Created
Latest
Jobs
Pre-run
Running
Holding
Finished
Failed
Cancelled
Merging
buildJob
Site

Real-life examples

Bit Streams

- Common in device output
- Organized in bunch of hex “words” of fixed length (*e.g.* 8 words = 32 bits)
- Often encapsulated as:



Bit Streams

- Need a:
  - **format**, often defined in a document
  - **encoder**, often on-board firmware
  - **decoder**, often a C++/FORTRAN program

Bit Streams

What	Format	Comment
Start Stream word Start Stream word	$0xC1A0F0F0$ $0xC1A0F0F1$	Fixed value for EDRO1 or Fixed value for EDRO2
Event counter and trigger flags	$0xXXXXXXYZ$	Progressive Event built counter (bits 31-10) and extended trigger flags (bits 9-0)
BCO counter	$0xXXXXXXXX$	Long format BCO counter at the build time
Clk counter	$0xXXXXXXXX$	Internal 40 MHz clock counter
Trigger Data	$0xA0A000TS$	Trigger information. Bits 31-8 fixed to "0xA0A000" Bits 7-5 fixed to 0 Bits 4-0, Time stamp of the event
AM Data	$0xXXXXXXXX$	Associative memory data as received (N_{AM} words)
Hit blocks Up to 8 blocks, one per layer ID: bits 26-24	$0xDDHHHHHH$	Hits from a given layer (N_i words) Bits 31-27 fixed to $1B$, Bits 26-24 identifier of the input line (0-7) Bits 23-0 Hits as received from the FED Last word in the block is an End-Event
End Stream word	$0xB1EB1E0F$	Fixed value
Check word	$0xXXXXXXXX$	The XOR of all words in an event including this should be 0

Table 2: Format of the EDRO event: a complete event include the Start Stream word, 3 counters, 1 trigger flag, 1 trigger data block, 8 hit blocks, the End Stream word and a check word, for a minimum of 15 words for pure empty events.

HEADER

device info

raw data

FOOTER

```
53) 1100 1100 0001 0010 0011 0100 1100 1100  cc1234cc     Event header
54) 0000 0000 0000 0000 0000 0000 0111 0010  00000072     +
55) 0000 0000 0000 0000 0000 0000 0000 1011  0000000b     +
56) 0000 0011 0000 0001 0000 0000 0000 0000  03010000     +
57) 0000 0000 0000 0000 0000 0000 0000 0001  00000001     +
58) 0000 0000 0000 0000 0000 0000 0000 0001  00000001     +
59) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     +
60) 0000 0000 0000 0000 0000 0000 0000 0011  00000003     +
61) 0000 0000 0000 0000 0000 1100 0001 1000  00000c18     + Run Number 3096
62) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     +
63) 0000 0000 0010 1011 0100 1011 0101 0010  002b4b52     +
64) 1101 1101 0001 0010 0011 0100 1101 1101  dd1234dd     + ROB header
65) 0000 0000 0000 0000 0000 0000 0010 1101  0000002d     + *
66) 0000 0000 0000 0000 0000 0000 0000 1010  0000000a     + *
67) 0000 0011 0000 0001 0000 0000 0000 0000  03010000     + *
68) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     + *
69) 0000 0000 0000 0000 0000 0000 0000 0011  00000003     + *
70) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     + *
71) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     + *
72) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     + *
73) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     + *
74) 1110 1110 0001 0010 0011 0100 1110 1110  ee1234ee     + * ROD header
75) 0000 0000 0000 0000 0000 0000 0000 1001  00000009     + * Format version 9
76) 0000 0011 0000 0001 0000 0000 0000 0000  03010000     + *
77) 1110 1101 1010 0000 0011 0000 0000 0000  eda03000     + * Source ID: Slave
78) 0010 0010 0010 1001 0110 1110 1000 1111  22296e8f     + * BX: 573140623
79) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     + * Lvl1ID: 0
80) 0000 0000 0010 1011 0100 1011 0101 0010  002b4b52     + * BCO: 2837330
81) 0001 0000 0000 0001 0000 1011 0001 1010  10010b1a     + * Trigger type: EXT     TS: 26
82) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     + * Event type: 0
83) 0000 0000 0000 0011 0000 0000 0000 0010  00030002     + * N Hit Words 0-3:      (3, 1, 4, 1, )
84) 0000 0000 0000 0000 0000 0000 0000 0000  00000000     + * N Hit Words 4-7:      (1, 1, 1, 1, )
85) 1010 0000 1010 0000 0000 0000 0001 1010  a0a0001a     + * Start trigger   0 trks End Trigger
86) 1101 1000 0000 0000 0000 0010 0000 0000  d8000200     + * Hit Block for line 0
87) 1101 1000 0001 1010 1011 1001 0001 0100  d81ab914     + * Hit Block for line 0
88) 1101 1000 0001 1010 1010 0101 0010 1001  d81aa529     + * Hit Block for line 0
89) 1101 1001 0000 0000 0000 0000 0000 0000  d9000000     + * Hit Block for line 1
90) 1101 1010 0000 0000 0000 0011 0000 0000  da000300     + * Hit Block for line 2
91) 1101 1010 0001 1010 1010 1001 1000 0000  da1aa980     + * Hit Block for line 2
92) 1101 1010 0001 1010 1011 1001 1000 0010  da1ab982     + * Hit Block for line 2
93) 1101 1010 0001 1010 1010 0101 0010 1001  da1aa529     + * Hit Block for line 2
94) 1101 1011 0000 0000 0000 0000 0000 0000  db000000     + * Hit Block for line 3
95) 1101 1100 0000 0000 0000 0000 0000 0000  dc000000     + * Hit Block for line 4
96) 1101 1101 0000 0000 0000 0000 0000 0000  dd000000     + * Hit Block for line 5
97) 1101 1110 0000 0000 0000 0000 0000 0000  de000000     + * Hit Block for line 6
98) 1101 1111 0000 0000 0000 0000 0000 0000  df000000     + * Hit Block for line 7
99) 1001 1010 1111 1111 1111 1111 1111 1111  9affffff     + *
100) 1011 1010 1111 1111 1111 1111 1111 1111 baffffff     + *
101) 1101 1010 1111 1111 1111 1111 1111 1111 daffffff     + *
102) 1111 1010 0000 0000 0000 0000 0000 0000 fa000000     + *
103) 1011 0001 1110 1011 0001 1110 0000 1111 b1eb1e0f     + * End ROD
104) 1111 1001 0001 1101 0110 0011 1101 1110 f91d63de     + * End ROB;  Checksum ok
```

Higgs Challenge

https://www.kaggle.com/c/higgs-boson

- Use the ATLAS Experiment to identify the Higgs boson!
- Machine learning challenge: register, download the data, run you AI program, upload the result, win $7000
- Separate signal from background
- Program providing better separation (AMS) wins

$$\text{AMS} = \sqrt{2\left((s+b+b_r)\log\left(1+\frac{s}{b+b_r}\right)-s\right)}$$

Higgs Challenge

- training: 250,000 events with ID + 30 features
- test: 500,000 with ID + 30 features
- Run on training, then on test

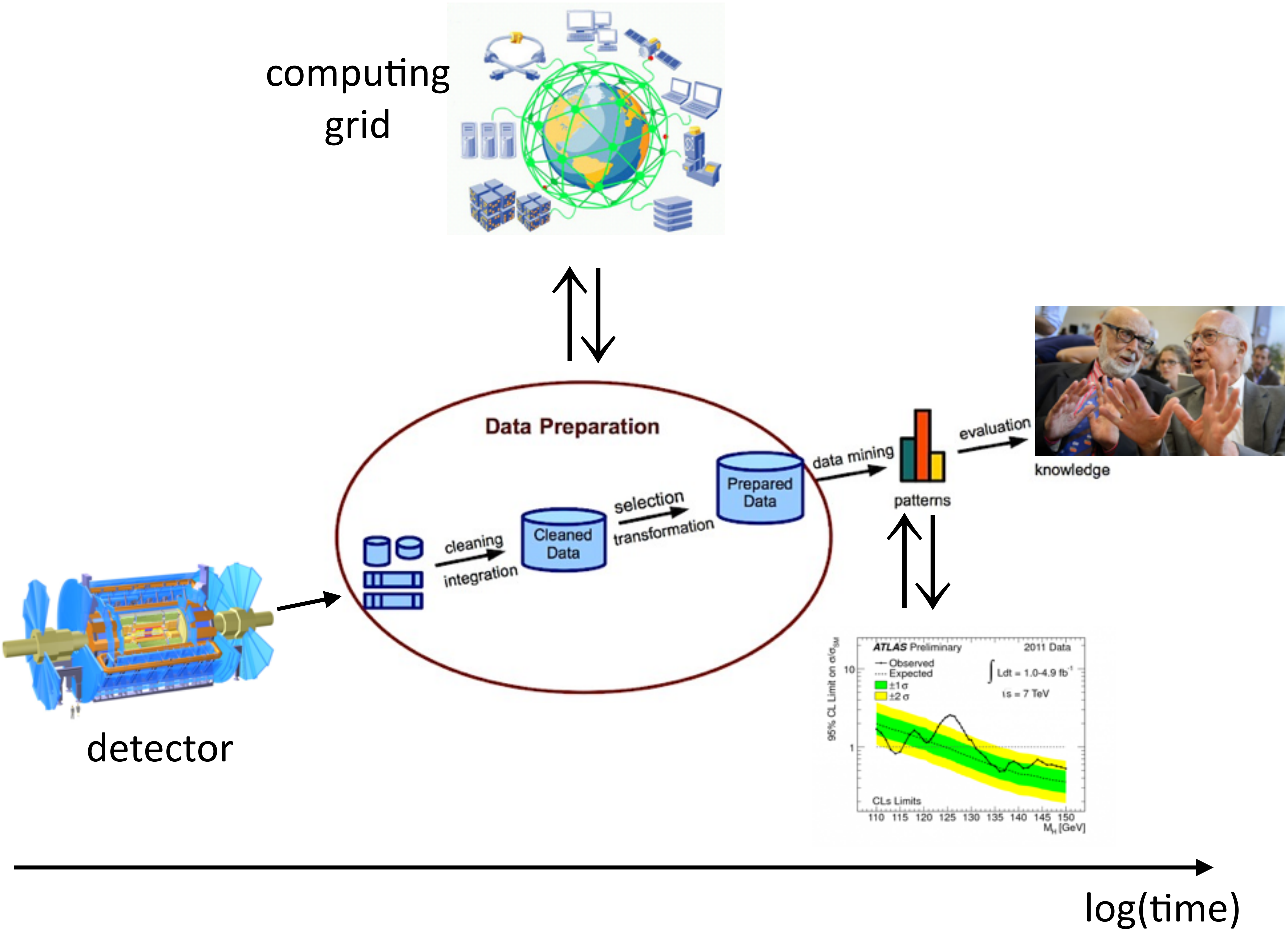
```
EventId,DER_mass_MMC,DER_mass_transverse_met_lep,DER_mass_vis,DER_pt_h,DER_deltaeta_jet_
jet,DER_mass_jet_jet,DER_prodeta_jet_jet,DER_deltar_tau_lep,DER_pt_tot,DER_sum_pt,DER_pt
_ratio_lep_tau,DER_met_phi_centrality,DER_lep_eta_centrality,PRI_tau_pt,PRI_tau_eta,PRI_
tau_phi,PRI_lep_pt,PRI_lep_eta,PRI_lep_phi,PRI_met,PRI_met_phi,PRI_met_sumet,PRI_jet_num
,PRI_jet_leading_pt,PRI_jet_leading_eta,PRI_jet_leading_phi,PRI_jet_subleading_pt,PRI_je
t_subleading_eta,PRI_jet_subleading_phi,PRI_jet_all_pt
350000,-999.0,79.589,23.916,3.036,-999.0,-999.0,-999.0,0.903,3.036,56.018,1.536,-1.404,-
999.0,22.088,-0.54,-0.609,33.93,-0.504,-1.511,48.509,2.022,98.556,0,-999.0,-999.0,-999.0
,-999.0,-999.0,-999.0,-0.0
350001,106.398,67.49,87.949,49.994,-999.0,-999.0,-999.0,2.048,2.679,132.865,1.777,-1.204
,-999.0,30.716,-1.784,3.054,54.574,-0.169,1.795,21.093,-1.138,176.251,1,47.575,-0.553,-0
.849,-999.0,-999.0,-999.0,47.575
bla bla bla
```

Higgs Challenge

- submission: see the evaluation page for details: https://www.kaggle.com/c/higgs-boson/details/evaluation

```
EventId,RankOrder,Class
1,2,b
2,541234,s
3,5,b
4,1,b
5,542456,s
...
```

Data Preparation in HEP


computing grid
Data Preparation
cleaning
integration
Cleaned Data
selection
transformation
Prepared Data
data mining
patterns
evaluation
knowledge
ATLAS Preliminary
2011 Data
Observed
Expected
CLs Limits
√s = 7 TeV
detector
log(time)

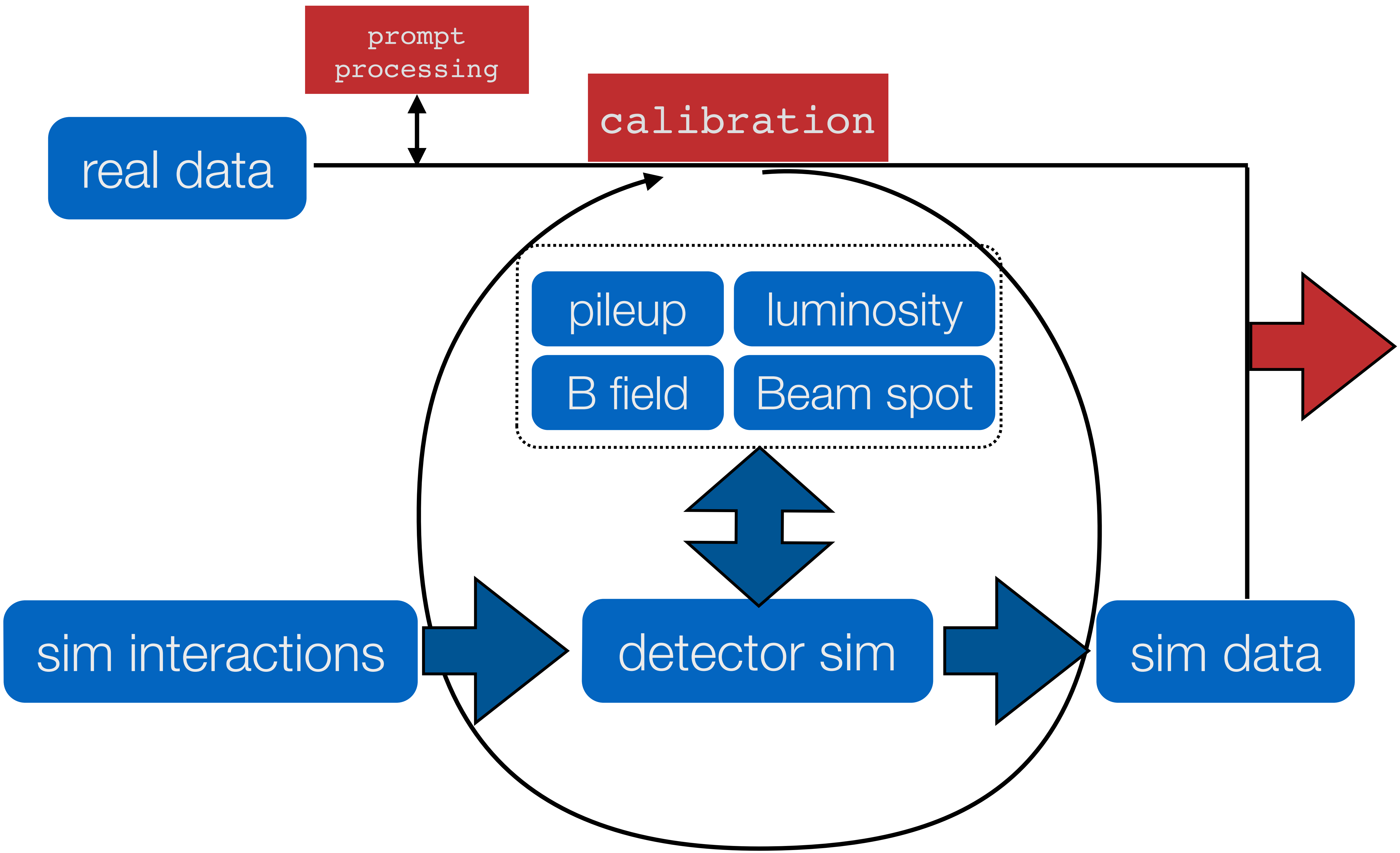
many aspects have to be covered!

- Run Conditions
- Data Quality
- Calibration
- Magnetic Field
- Detector Simulation
- Non-Collisions Background
- Reprocessing
- Prompt Reconstruction
- Event Display
- Luminosity
- Run and Data Information
- Beam Spot
- Analysis Facility

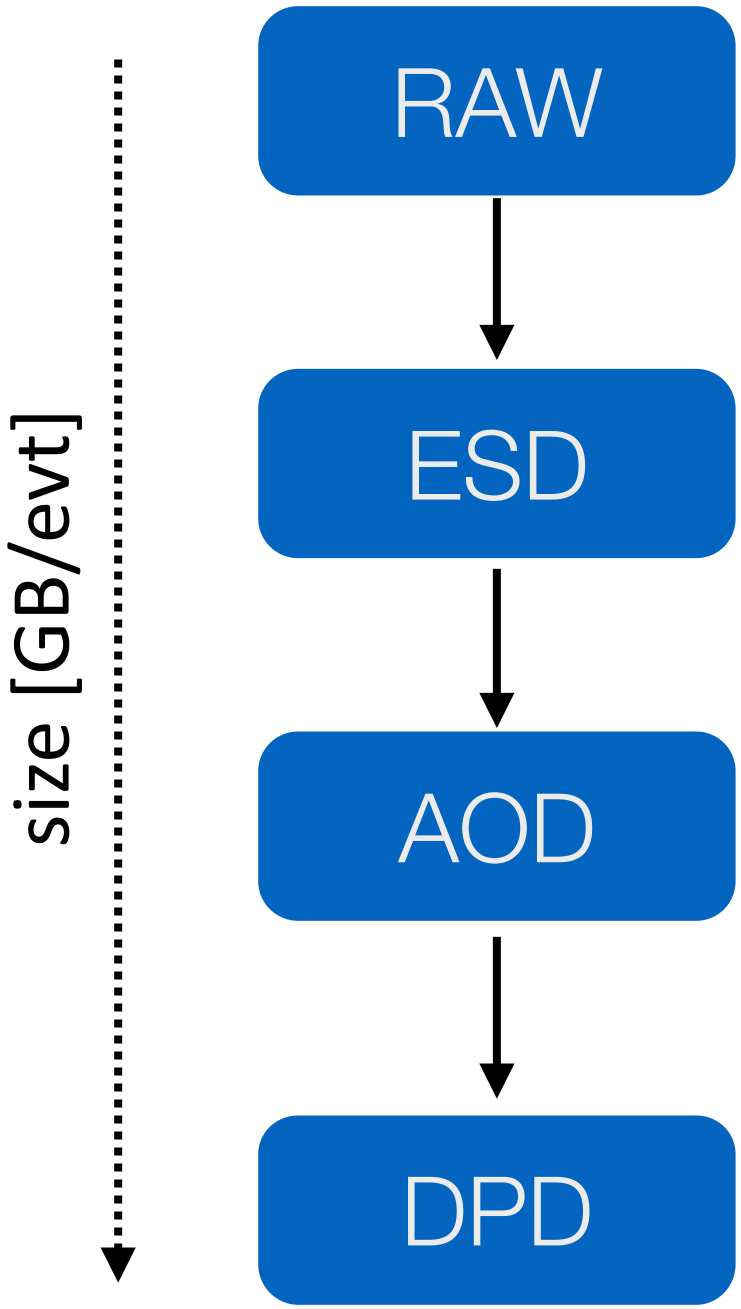
many aspects have to be covered!

- Run Conditions
- Data Quality
- Calibration
- Magnetic Field
- Detector Simulation
- Non-Collisions Background
- Reprocessing
- Prompt Reconstruction
- Event Display
- Luminosity
- Run and Data Information
- Beam Spot
- Analysis Facility

Data Preparation Cycle



Data distillery (ATLAS)



raw data, detector output.
waveforms, integrated charge, etc..

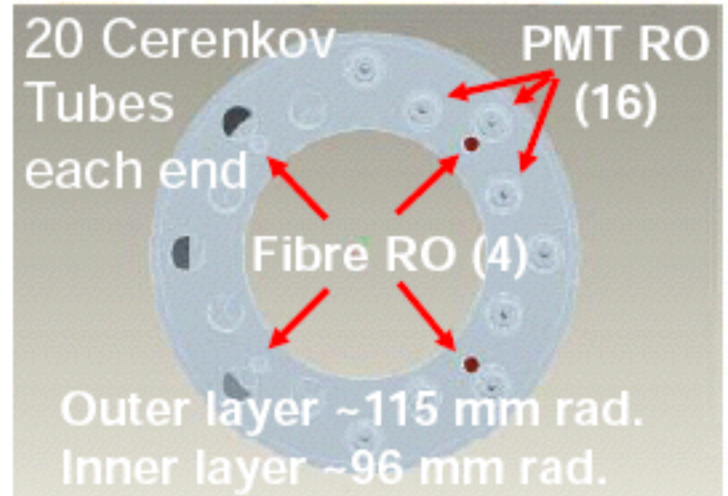
Full-fledged event reconstruction.
Calo cells, tracks, electrons, muons, jets

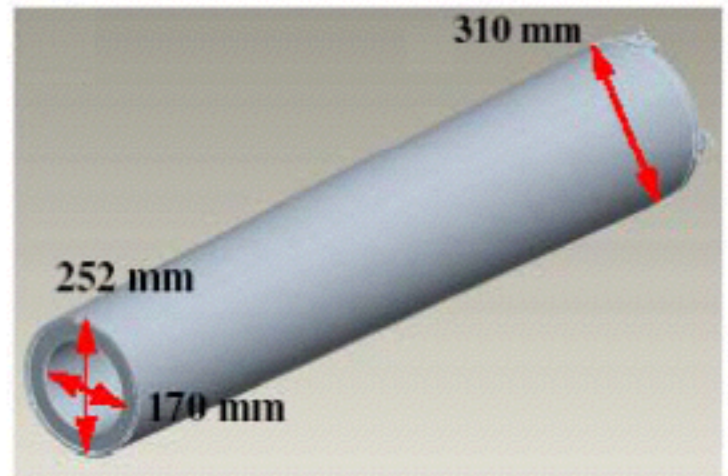
Analysis-oriented data. Contains only information needed for physics. *Arrays of structures* (e,mu,jet)

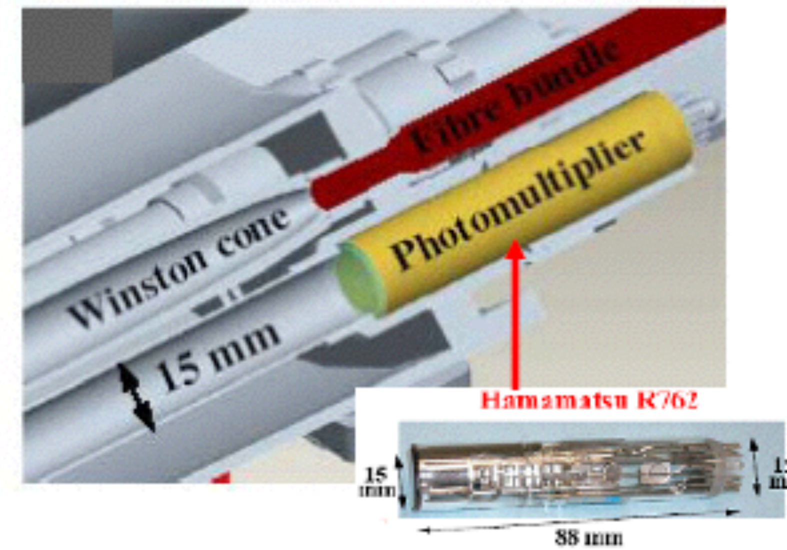
End-user-oriented data. Contains only information needed for specific analyses
Structure of arrays

LUCID’s PMTs calibration

- Cherenkov light detector for online luminosity
- Inelastic pp collisions in the forward direction
- Measure integrated and instantaneous luminosity and beam conditions







- 20 aluminum tubes filled with air (was: C_4F_{10})
- Two twin detectors placed at both sides of ATLAS pointing towards the interaction point
- Covering $5.61 < |\eta| < 5.93$
- Light collected by photomultiplier tubes (PMTs)
- More info: https://twiki.cern.ch/twiki/bin/view/Atlas/LucidDescription

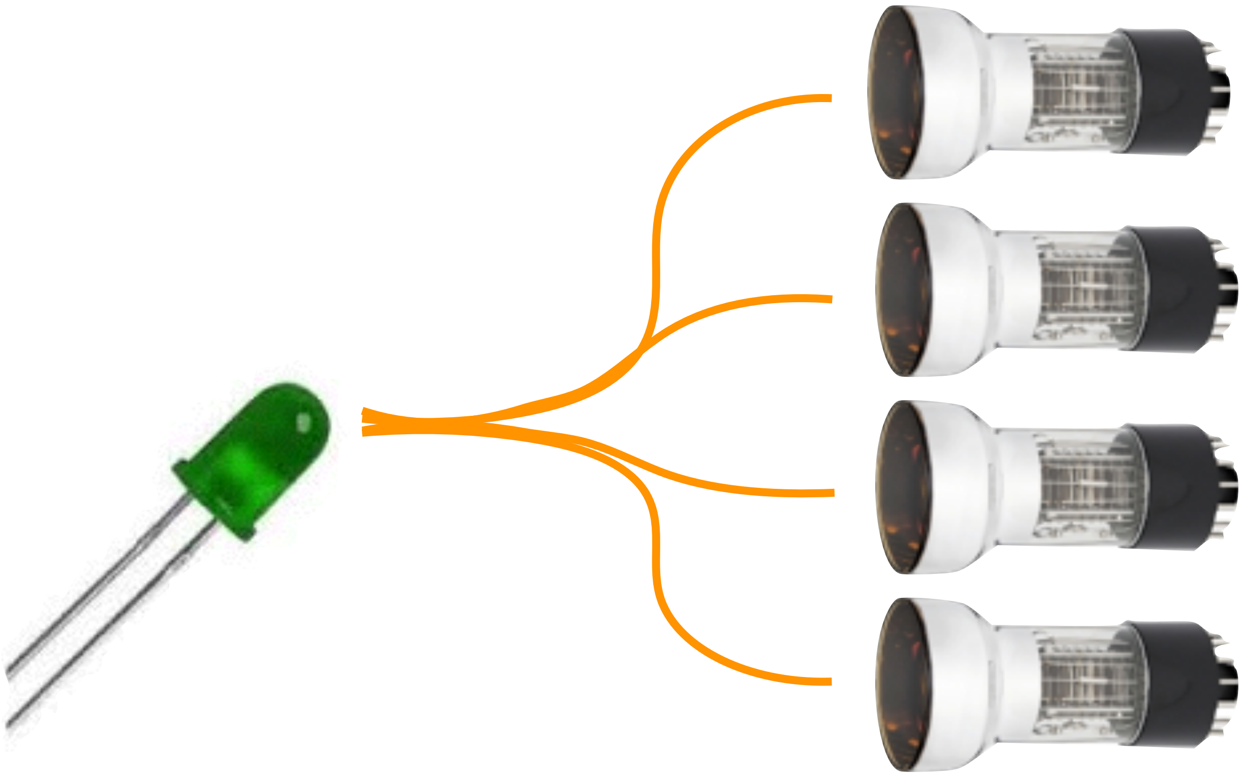
in a nutshell

- Cherenkov rad. → PMT signal → discriminator → hits
- Collection of hits → Events
- Event rate ~ luminosity (Poisson distribution).
  Prop. factor calibrated using Van der Meer scan
- Several algorithms devised to provide luminosity per LB

PMT Calibration

Calibration = keep PMT gain constant

LED → optical fibers → PMTs

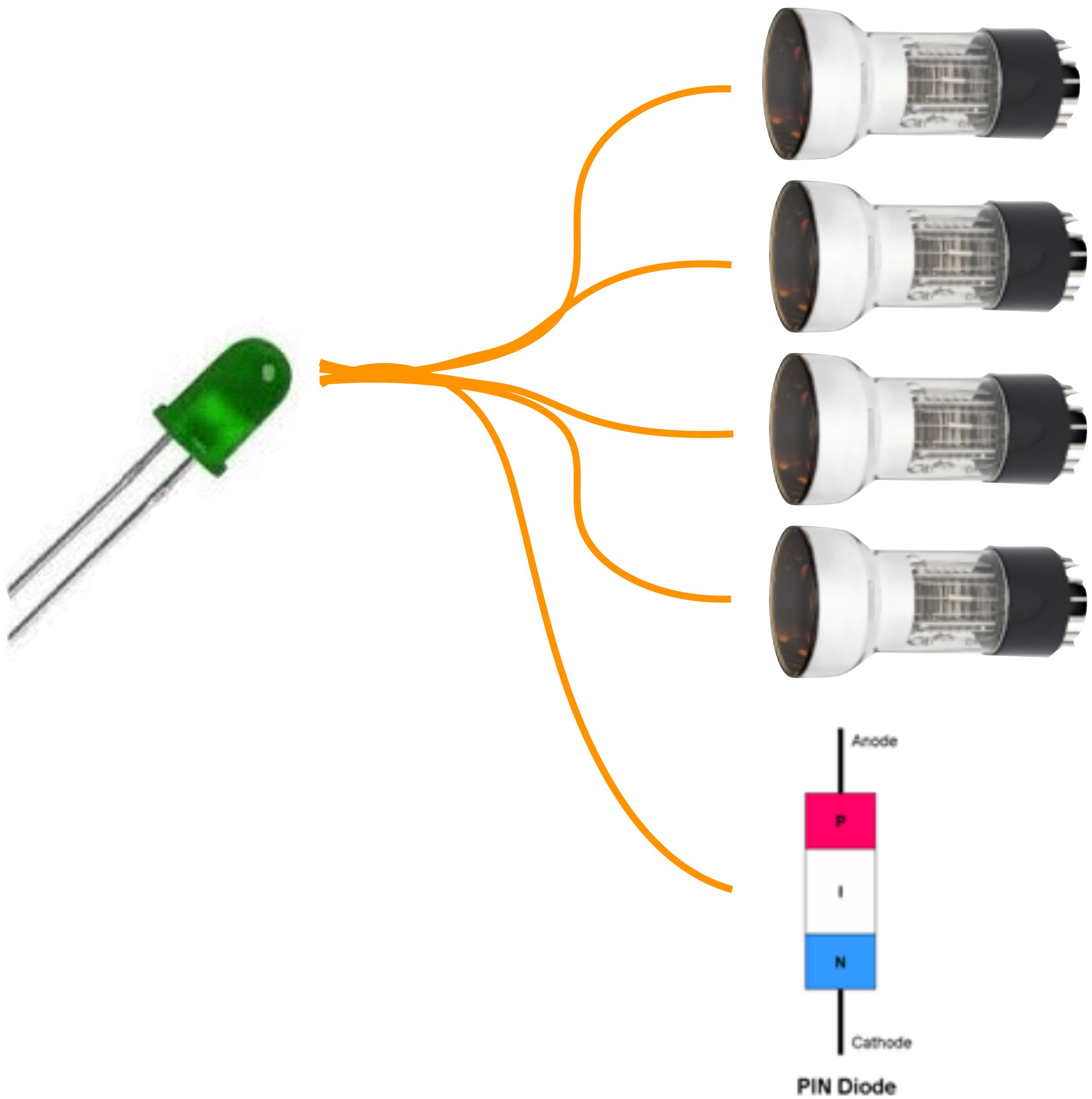


- photocathode aging
- fibers not rad-hard
- LED fluctuates

PMT Calibration

Calibration = keep PMT gain constant

LED → optical fibers → PMTs



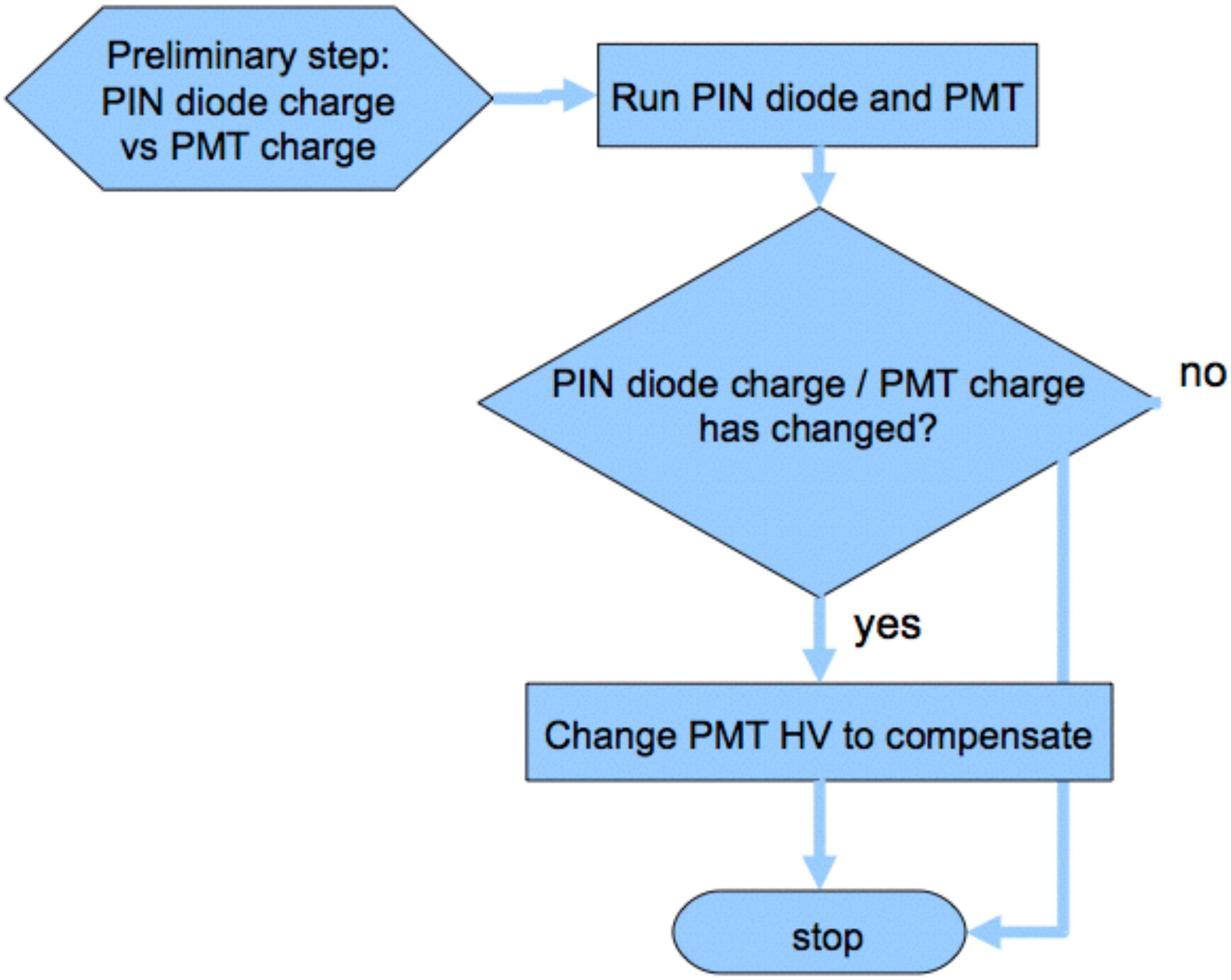
- photocathode aging
- fibers not rad-hard
- LED fluctuates

PIN diode (very stable, rad-hard)

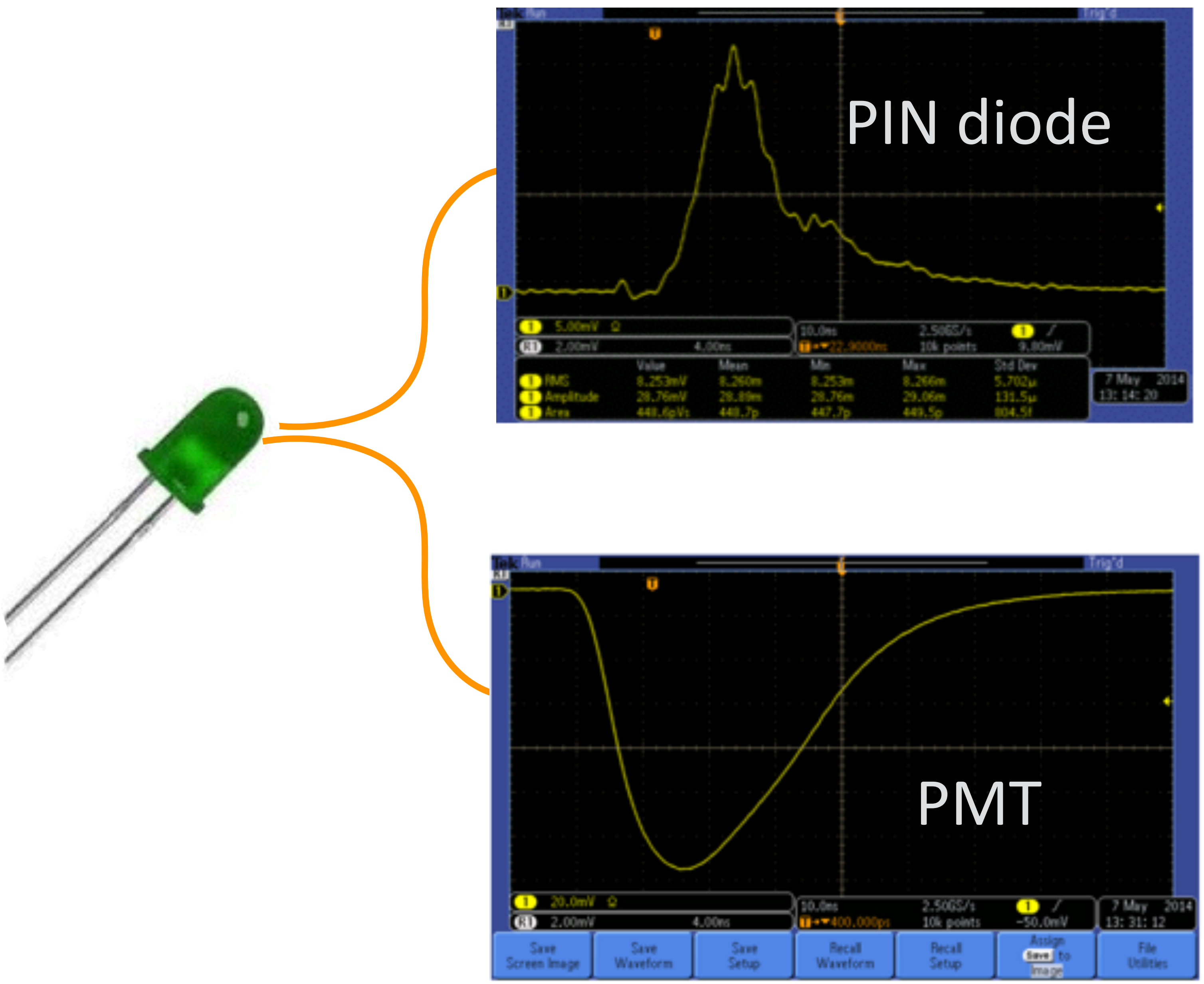
⇒Compare int'd charges

PIN diode *vs* PMTs

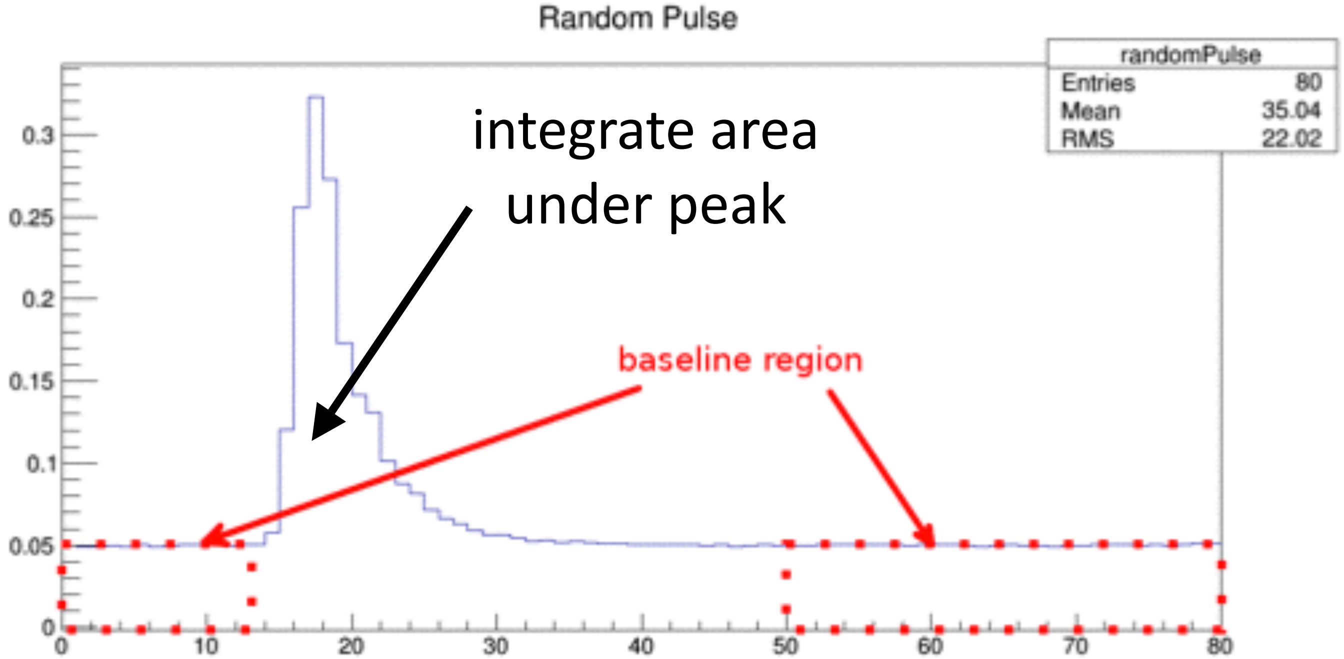
PMT Calibration

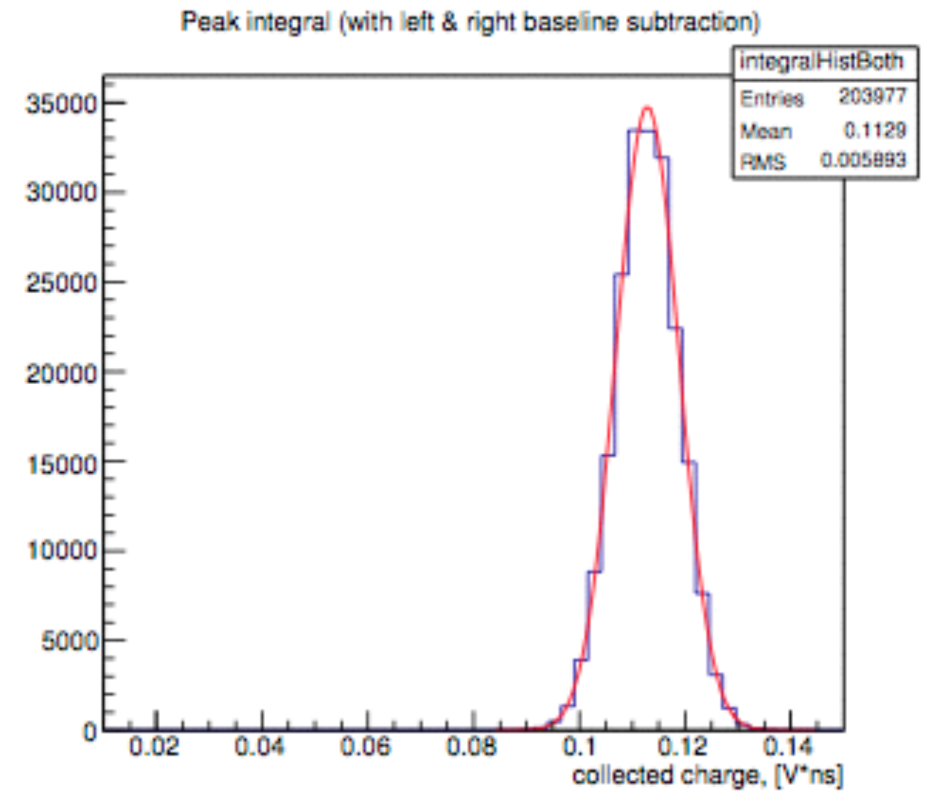


PMT Calibration

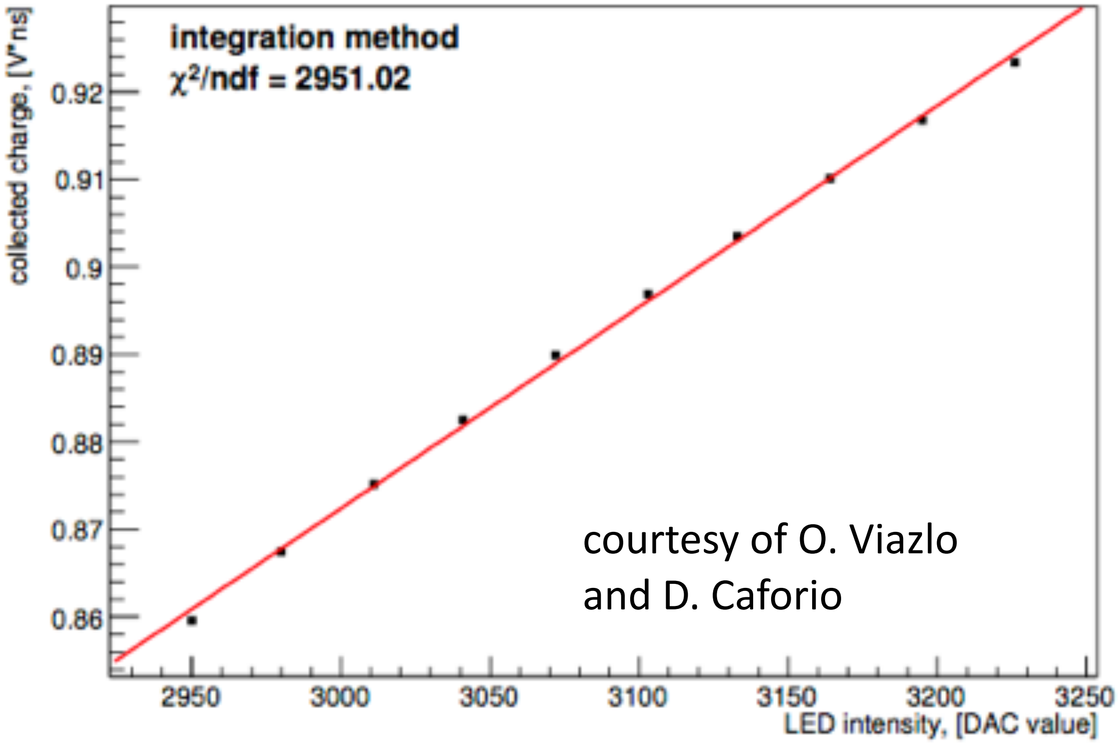


PMT Calibration





- integrate pulse: Q_{pulse}
- integrate baseline: Q_{BL}
- calculate charge in peak: $Q_{peak} = Q_{pulse} - Q_{BL}\frac{80}{n_{BL}}$
  - total number of bins in pulse: 80 bins
  - n_{BL} - number of bins used to measure baseline
- fit charge distribution with gaussian; take mean as nominal value and sigma as error



Under the trunk

- TDAQ sw drives the LED, reads, stores and moves the data taken from the FADC
- PMT current can be read from the FADC of by the DCS power supply
  - Implemented in Siemens' SIMATIC WinCC Open Architecture (was: PVSS) (proprietary sw)
  - Data are stored in "data points" and moved around by the sw infrastructure

Under the trunk

- During calibration:
  - Each event is compressed into a binary files containing histograms, numerical values, etc.
  - TDAQ sw moves those files to rack computers
  - Files are unpacked to ROOT files by a cronjob
  - A script reads those files and plain txt files containing information about the int’d charges
  - A program is run to calculate the PMT/PIN ratio

That's all, folks

Bibliography

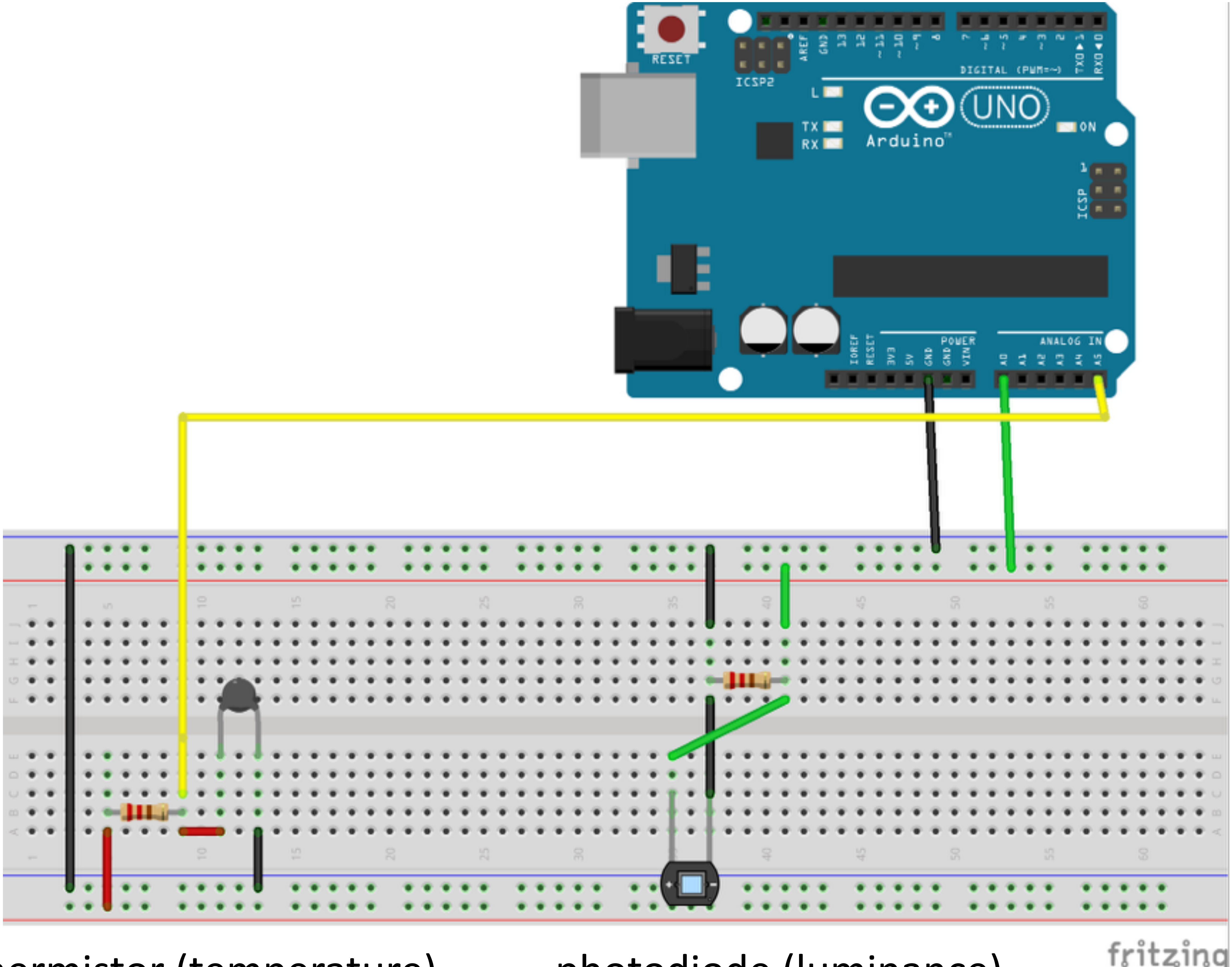
Most of these topics are "oral tradition" or described in handbooks. Suggested readings:

- O. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*, Springer
  http://www.cs.bme.hu/nagyadat/Data_Mining_and_Knowledge_Discovery.pdf
- A. Kuhlmann et al., *Data mining on Crash Simulation Data*, arXiv:cs/0505008v1
  http://arxiv.org/pdf/cs/0505008v1.pdf
- Data preparation in ATLAS:
  Improved luminosity determination in pp collisions at sqrt(s) = 7 TeV using the ATLAS detector at the LHC, arXiv:1302.4393v2

Hands-on exercise

Objectives

- Arduino board reads two sensors (luminance, temperature) and sends an output stream to a computer
- The output stream is published on a web page, generated on request
- Students have to decode the stream w/ sanity check, create and store a ROOT tree
- Then fill histograms, fit distributions, quote final values
- All materials on GIT hub: https://github.com/rdisipio/HASCO2014



thermistor (temperature) photodiode (luminance)

Data format

- A stream consists of a series of bits, grouped as four 8-bits words (32 bits in total)
- A stream must start with a `0xc1a0c1a0` (header) marker
- After the header, the number of events is stated: `0xa0a00000` + hex(number of events)
- For each event:
  - Event ID number: `0xe0000000` + hex(event ID)
  - Timestamp as the time in seconds since "the epoch" (01/01/1970) as a floating point number.
  - `0xd0000000` + 1000 times the Photodiode voltage (uint)
  - `0xd1000000` + 1000 times the Thermistor temperature (Celsius, uint)
- A stream must end with a `0xb1eb1e0f` (footer) marker
- Finally, the checksum represented by the XOR of all the words contained in the stream except the checksum. The XOR of all the words contained in the stream *including the checksum* must be zero.

`https://github.com/rdisipio/HASCO2014/blob/master/arduino4hasco2014_format.pdf`

A stream containing 5 events looks like this:

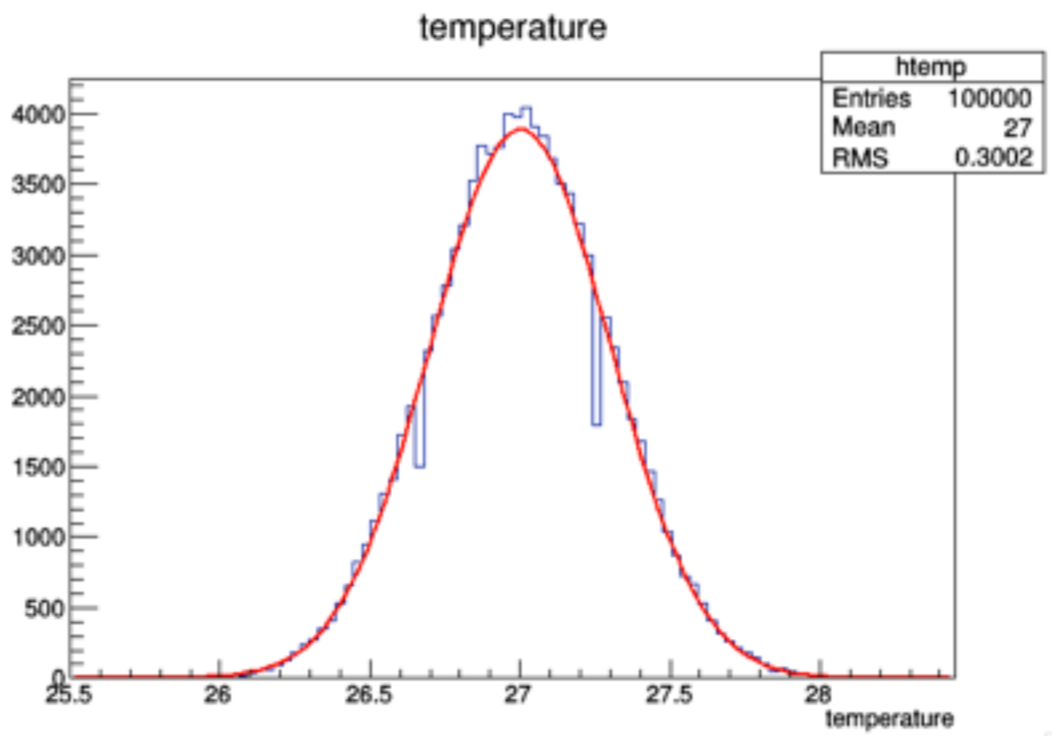
```
0xc1a0c1a0     header
0xa0a00005     number of events (5)
0xe0000000     event #1
0x53cfc699     timestamp
0xd0000c26     1000 * photodiode voltage
0xd1006b4e     1000 * thermistor temperature
0xe0000001     event #2
0x53cfc699
0xd0000c62
0xd1006ab8
0xe0000002     event #3
0x53cfc699
0xd0000c08
0xd10069a0
0xe0000003     event #4
0x53cfc699
0xd00000c8
0xd10069b4
0xe0000004     event #5
0x53cfc699
0xd0000082
0xd10068b0
0xb1eb1e0f     footer
0x62247c63     checksum
```

Your turn!

- Warm up with the simulated `stream.dat` (100k events)
- Then, try to read real data from the web server
  - http://0.0.0.0:5000/get_data

https://github.com/rdisipio/HASCO2014/blob/master/get_data.py

https://github.com/rdisipio/HASCO2014/blob/master/a_5_minutes_python_primer.pdf



- temperature/luminance correlated?
- Wrong entries? How do you react?