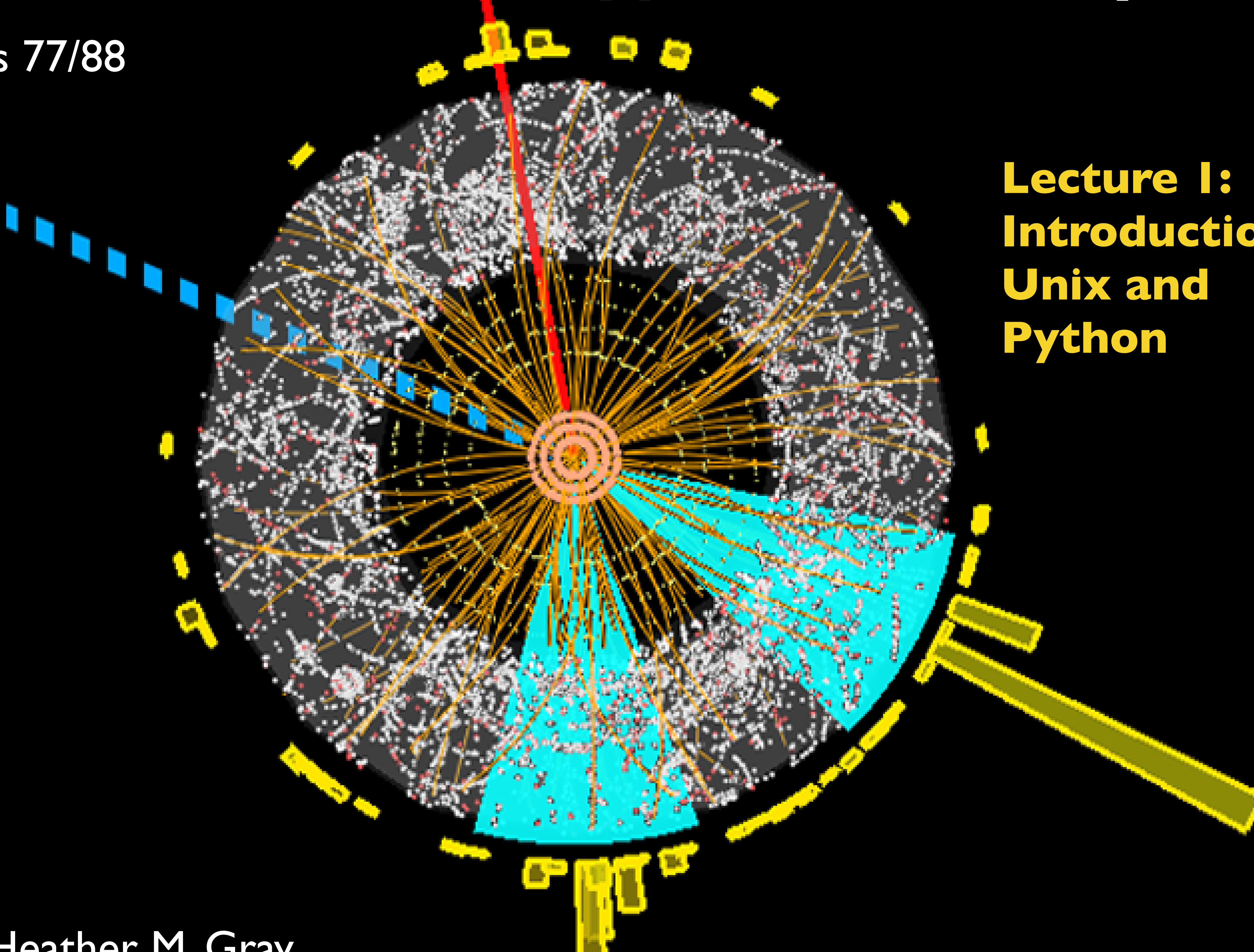
Introduction to Computational Techniques in Physics/Data Science Applications in Physics

Physics 77/88



Lecture 1: Introduction to Unix and Python

Prof. Heather M. Gray

Many thanks to Yury Kolomensky for material used in this course

The What and Why of Computing

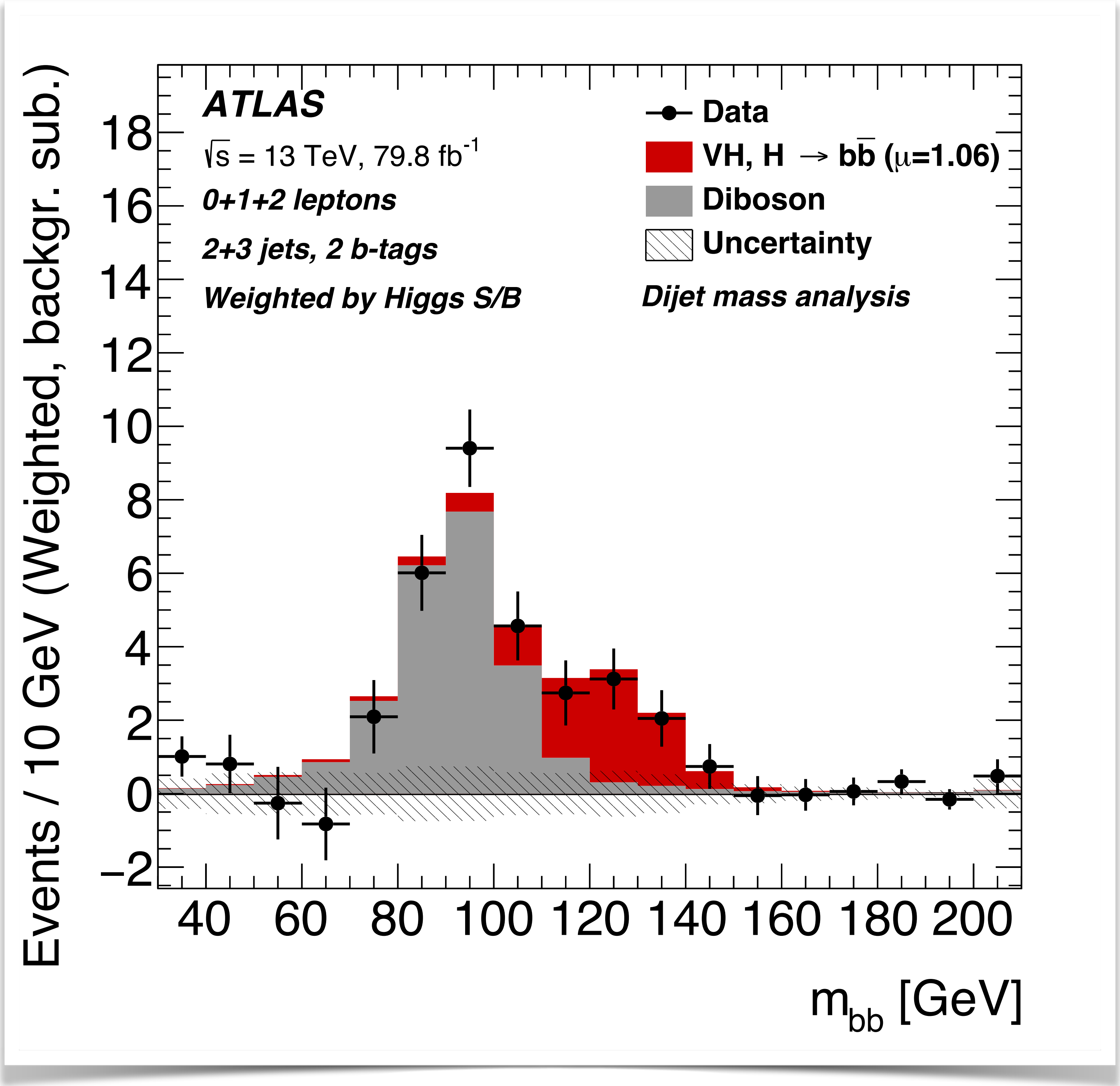
Why Computing?

- Necessary tool
  - Physics =
  - Observations →
    - Make a set of or
      - Summarize the
    - Most are drawn with some degree of
      - e.g.
      - In reality, we know to some precision
        - e.g. $G_N =$
  - Many measurements are (e.g.)
    - Have to be interpreted in

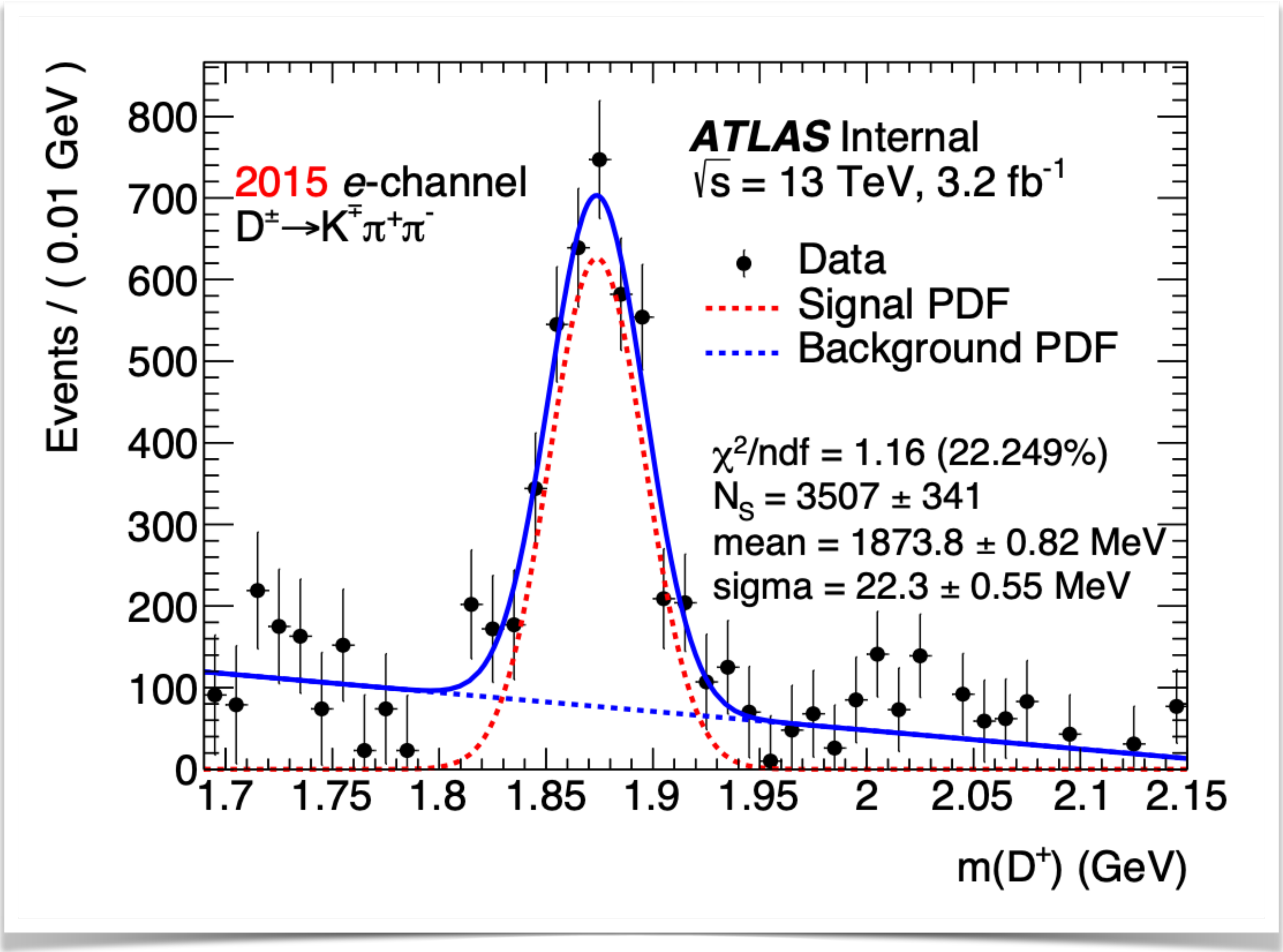
- Many measurements require of and complicated
  - Computerization
- are computerized
- And use computers too

My First Exposure to Computing

Examples (My Research)

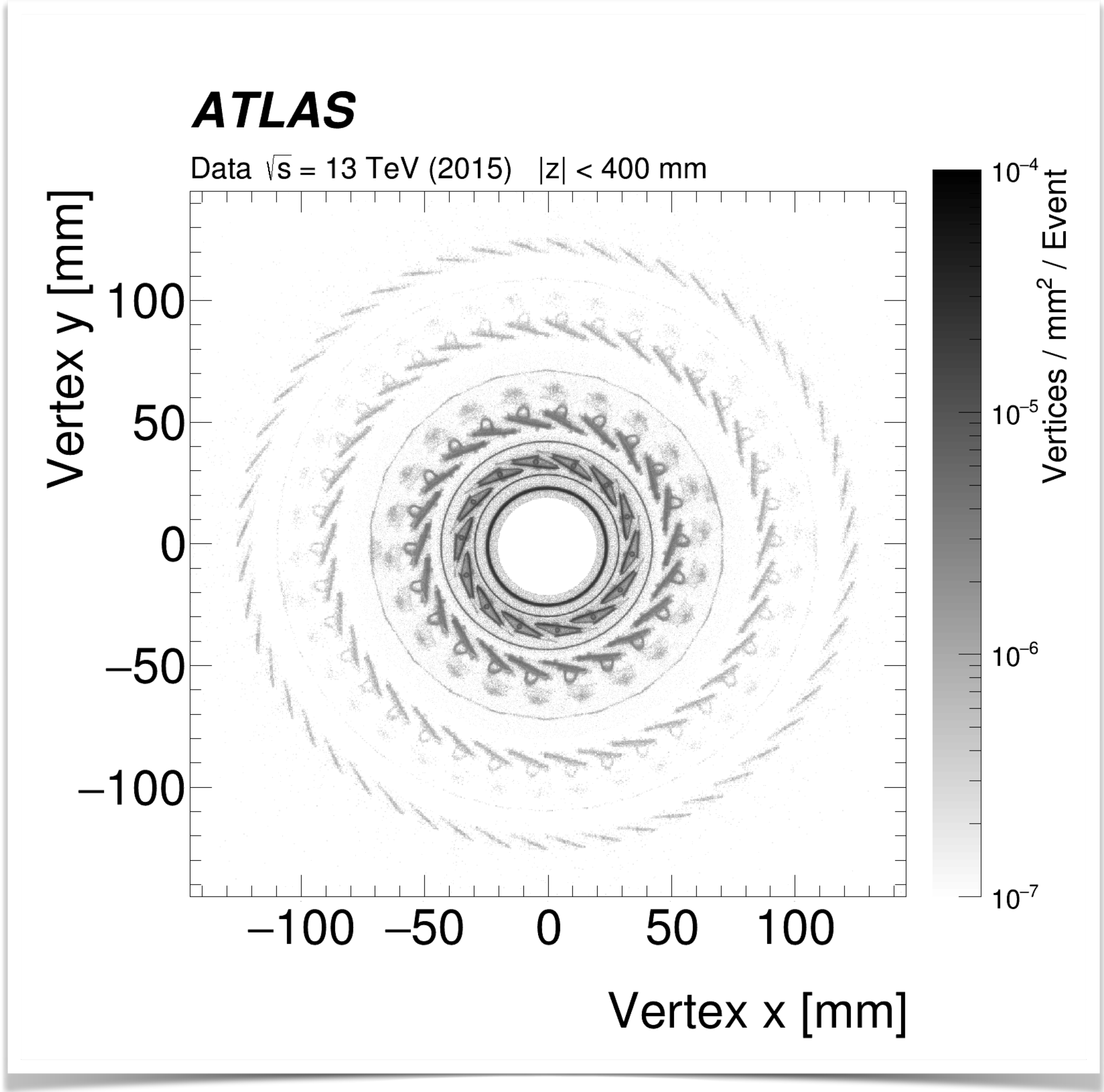


Examples (My Research)



M. Muskinja (PostDoc)

Examples (My Research)



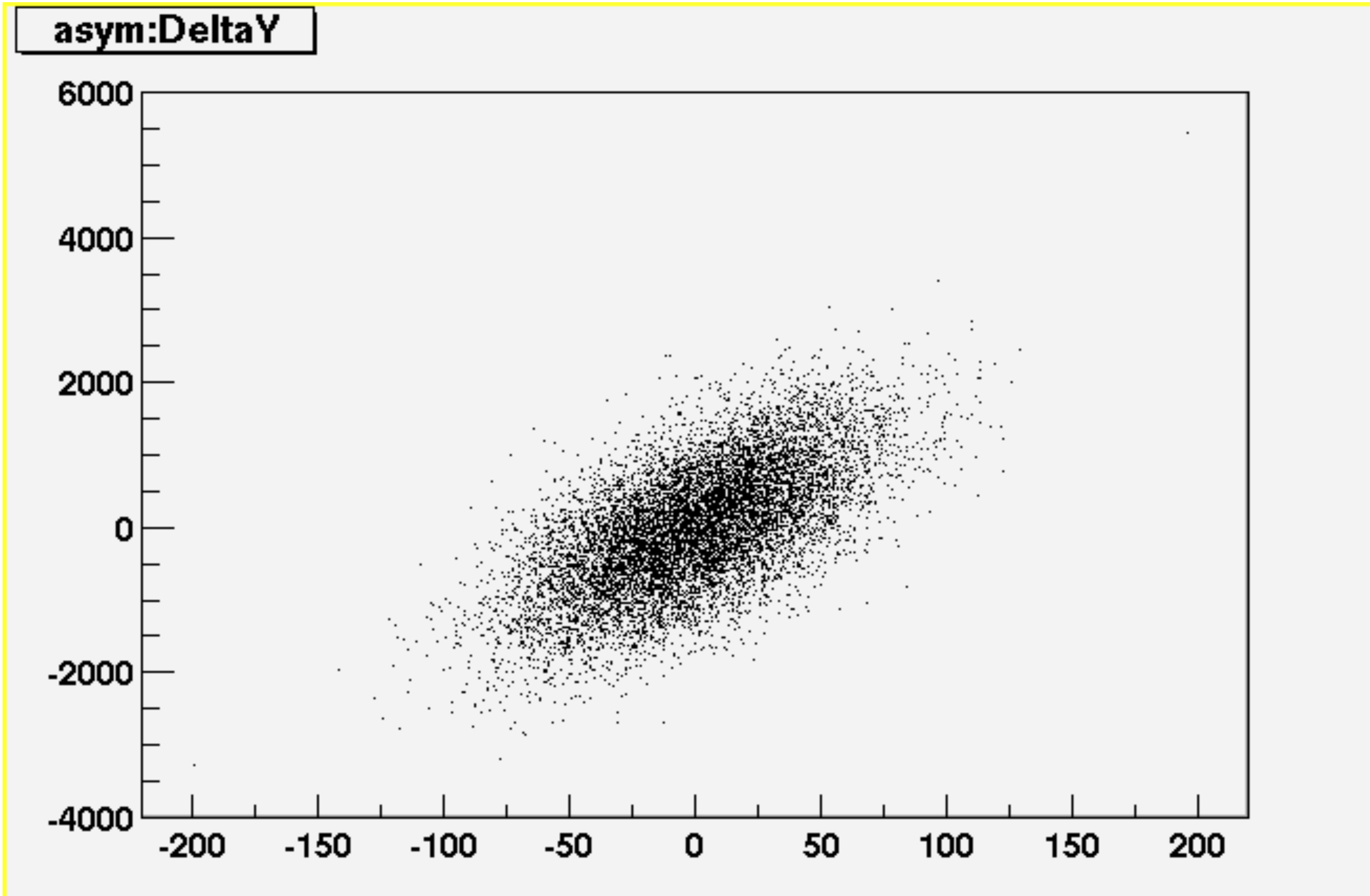
Describing the Data

- Data: results of
  - In physics we mostly deal with , i.e.
  - Other fields may deal with
    - An American Robin has gray upper parts and head, and orange under parts, usually brighter in the male
  - Numbers are to handle
    - We will mostly deal with
- Types of quantitative data
  - data, e.g.
  - data, e.g.
    - Some , e.g. from
  - data, e.g.
  - Sets of :

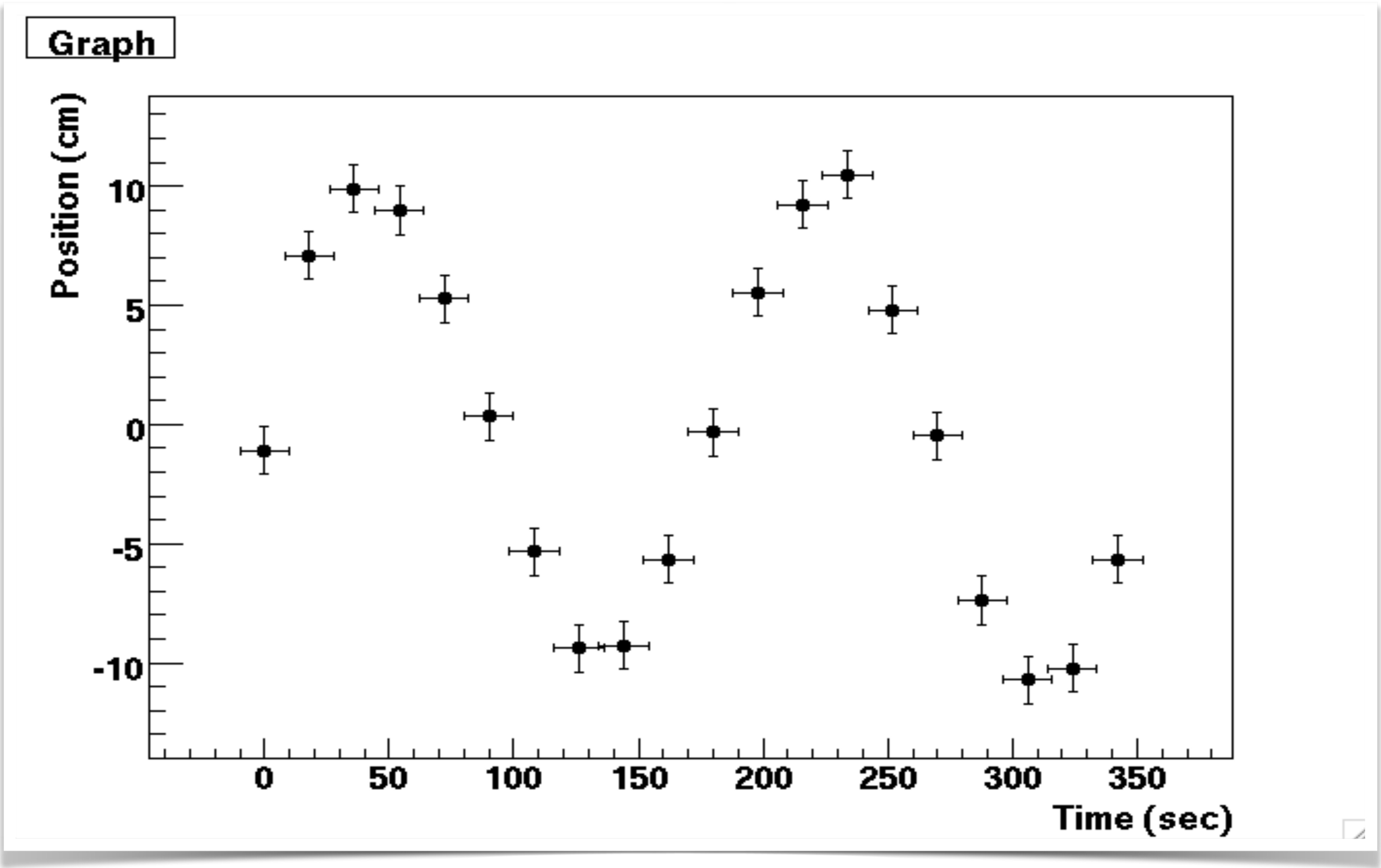
Visualizing the Data

- Tables
  - datasets can be difficult to
- Graphs
  -
- Charts
  -

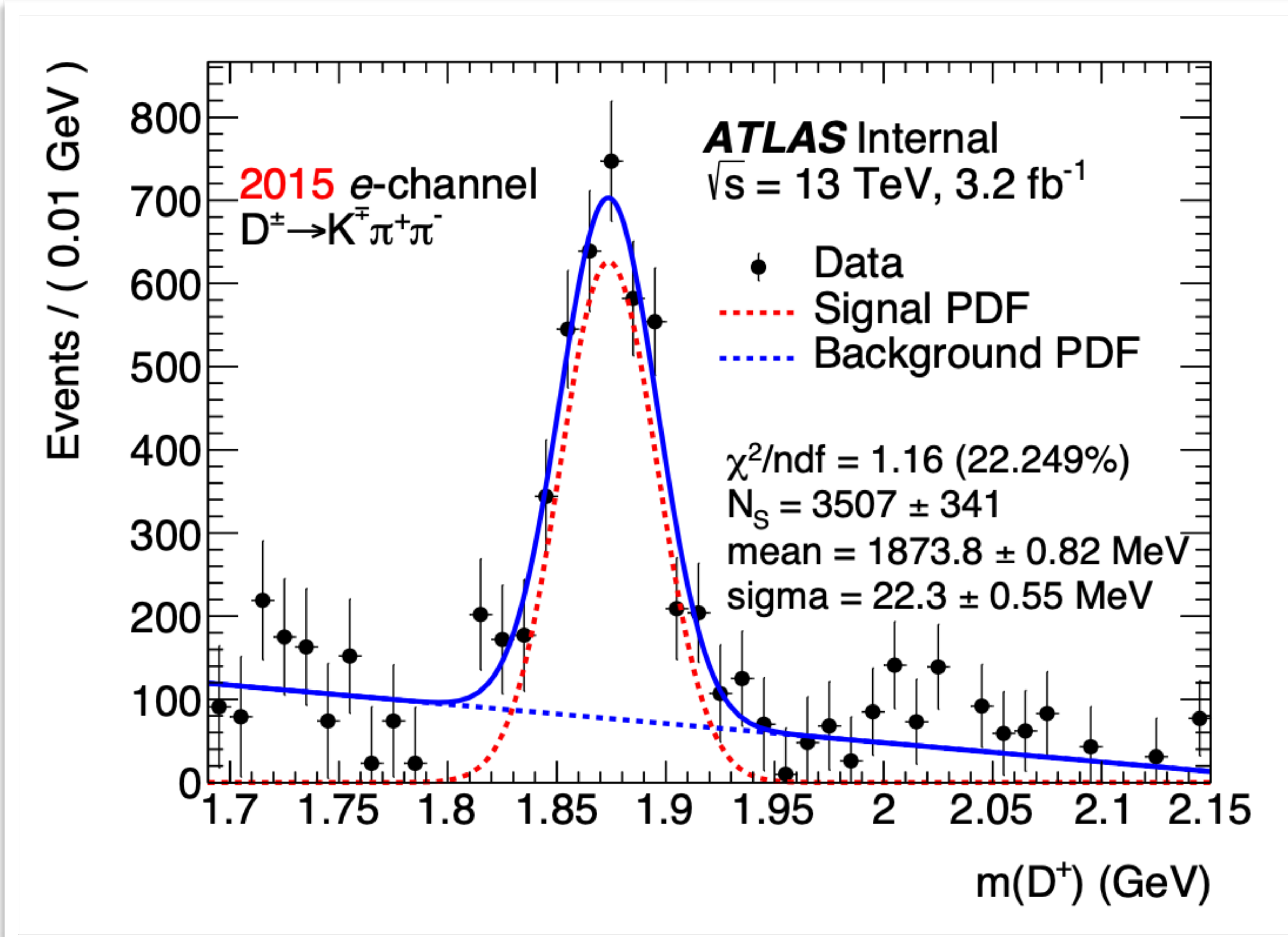
Examples


asym:DeltaY
6000
4000
2000
0
-2000
-4000
-200
-150
-100
-50
0
50
100
150
200

Examples

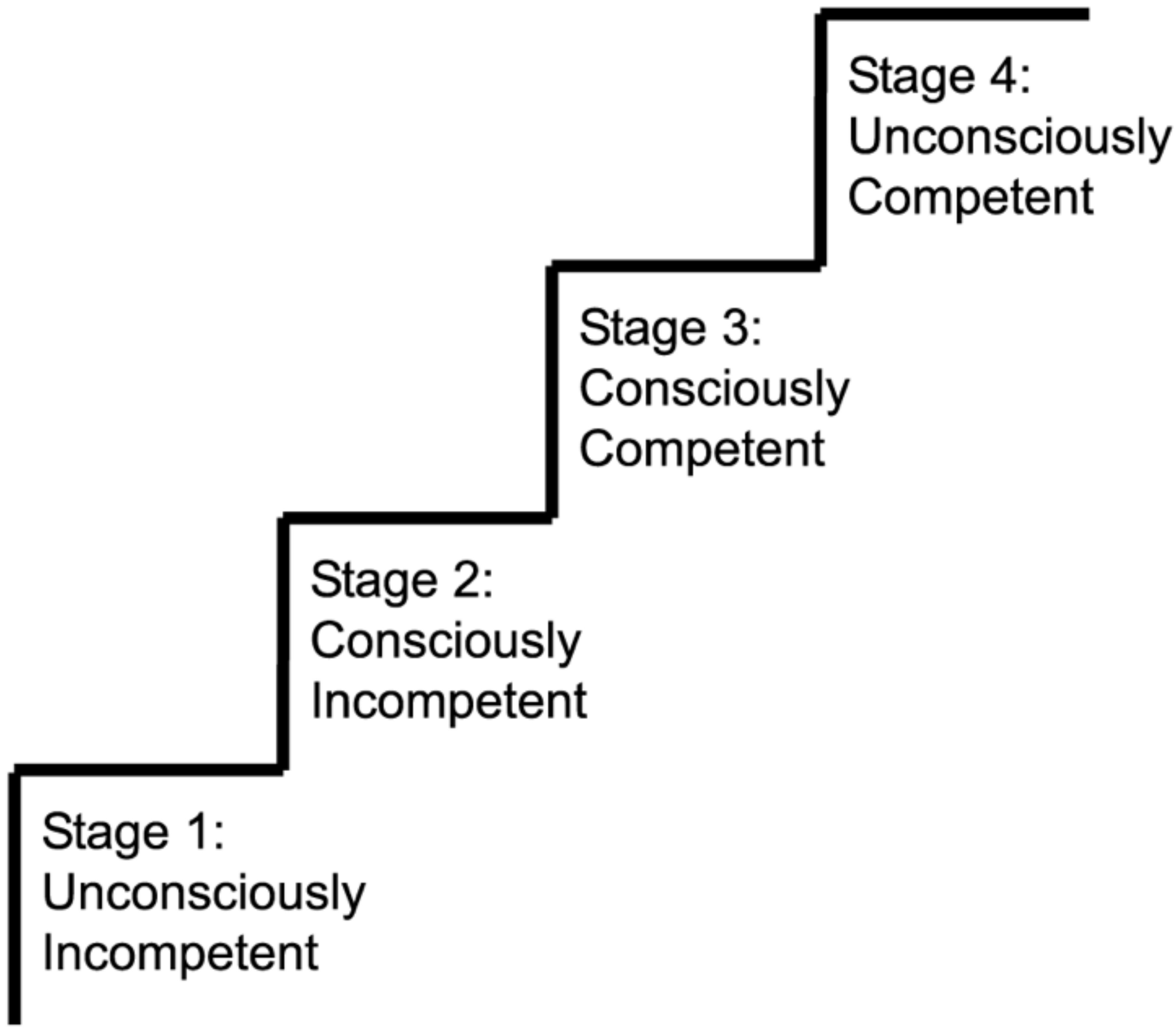

Graph
Position (cm)
10
5
0
-5
-10
0
50
100
150
200
250
300
350
Time (sec)

Examples



ATLAS Internal
√s = 13 TeV, 3.2 fb-1
2015 e-channel
D±→K∓π+π-
Data
Signal PDF
Background PDF
χ2/ndf = 1.16 (22.249%)
NS = 3507 ± 341
mean = 1873.8 ± 0.82 MeV
sigma = 22.3 ± 0.55 MeV
Events / (0.01 GeV)
m(D+) (GeV)

How You Should Learn



How to Approach This Course

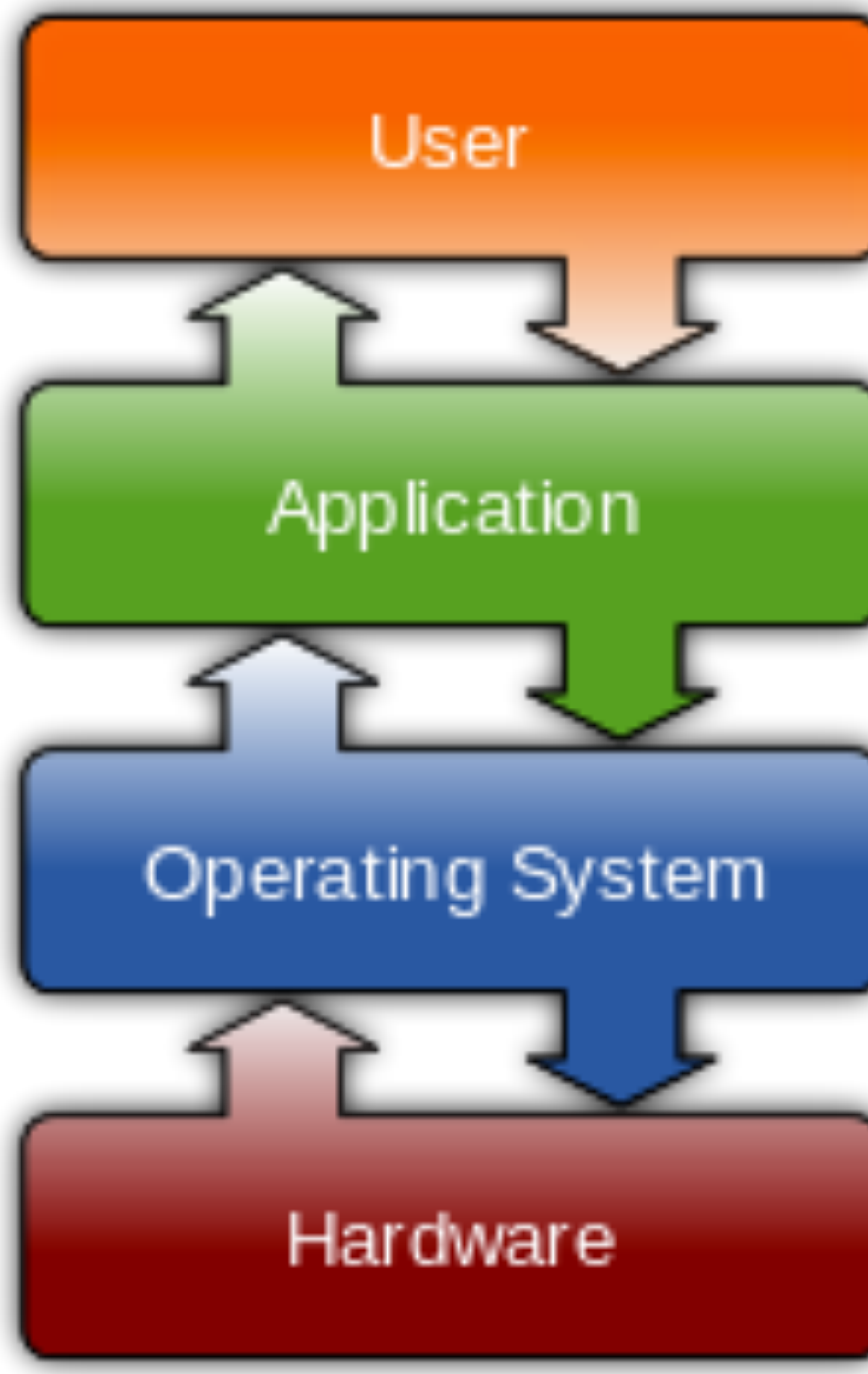
- Active Learning
  - Learn by
- Three main threads to the course
  -
  -
  -
- No single book covers all of it
  - Recommend two (Newman, Hughes & Hase)
  - Many online resources for each step
    - Book → Lecture → Workshop → HW → Book cycle applies

Let’s take a look at the jupyter notebook

Operating Systems, Programming Environments, Representations

Brief Spiel: Operating Systems

- Operating System (   ) is an interface between a   and a
  - Translate   to electronic signals, report results back
  - Optimized for hardware, efficient
    - Low-level code
  - Interface can be graphical (   ), text-based (   ), or mixed (   )



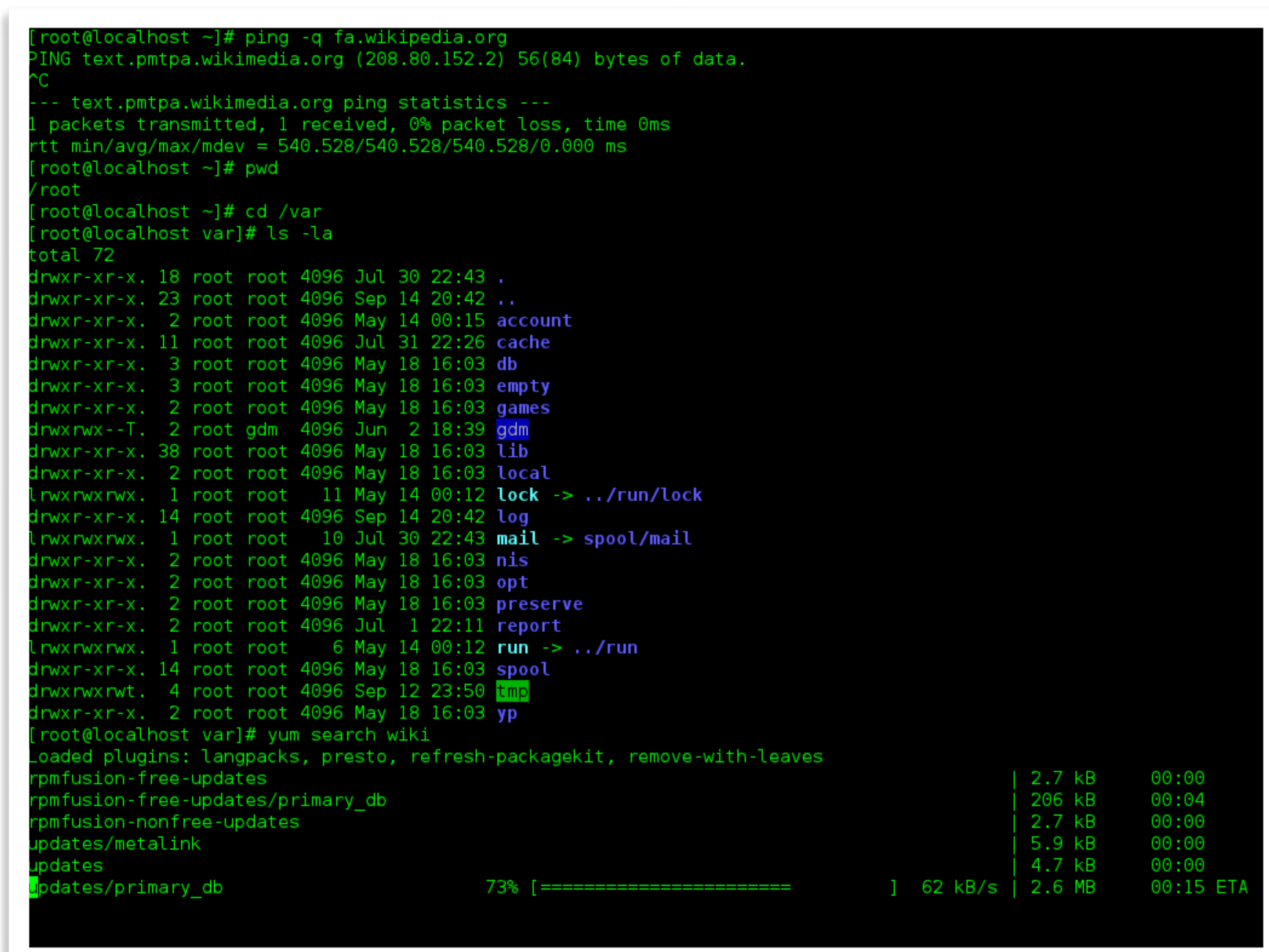
Source: Wikipedia

Unix/Linux

- One of the oldest (surviving) OS ( )
- Initially developed for , ported to (Linux and spinoffs)
- Variants are now running on a variety of hardware
    - PC ( )
    - Mac ( )
    - smartphones (,)
    - even the occasional microwave
- Robust, efficient
- Designed to be , so basic interface is a

Programming Environments

- interface
- interface
- Connecting to a server (ssh, terminal)
- Scripts
- Compiled vs interpreted languages



Source: Wikipedia

Brief Spiel: Programming Languages

- Programming languages allow to translate sets of (an algorithm) to a form understandable by a
- Classifications
  -
    -
  -
    -
- Implementations
  -
    -
  -
    -

Data Representation

- Data on computers represented in format
    - Base representation (as opposed to base)
        - $abcd_2$ =
    - Examples:

Decimal numbers	Binary equivalent	Decimal numbers	Binary equivalent
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

- Smallest memory cell:
    - bits = byte (B)
- Measures of memory
    - 1 kB = ; 1 MB = , etc

Binary Representation

- Practical consequences
  - All numbers in the digital format are , i.e. they have precision
    - This is easy to understand with (by construction)
    - Real numbers: think about representation in
      - 0.125 =
    - Could you represent 1/10 in powers of 2 ? What about π ?
  - Basic data types have and value, as well as precision, determined by the data type size
    - i.e. how much is allocated for each data type
    - Most common: or bytes for integer and real values

Examples: Integer Data

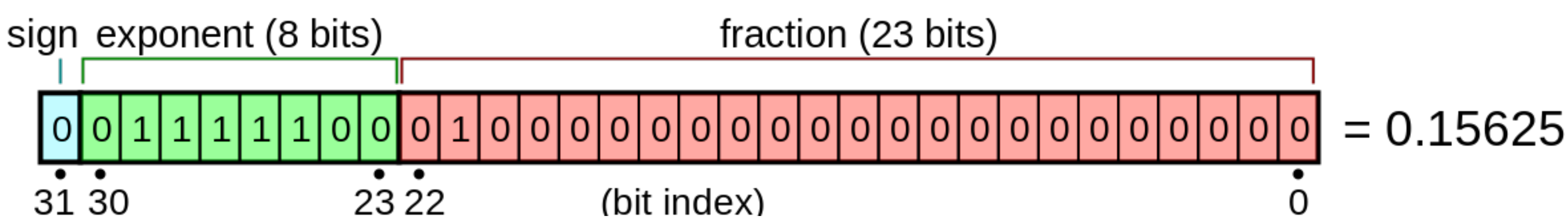
- C/C++

```
root [0] sizeof(int) // number of bytes for interger
(const int)4
root [1] sizeof(long) // number of bytes for long integer
(const int)8
root [2]
```

- This means:
  - Range of signed ints in C is ±2147483647
  - Range of unsigned ints in C is 0..4294967295
  - E.g. time in Unix is represented as — in seconds from Jan 1, . Hence the impending end of time in
    - (Google "end of unix time")

Examples: Real numbers

- Real numbers are represented by 3 fields
  -
  -
  -



$$\text{value} = (-1)^{\text{sign}} \times \left(1 + \sum_{i=1}^{23} b_{23-i} 2^{-i}\right) \times 2^{(e-127)}$$

- For example, most languages use and floating point numbers
  - and in C/C++

Precision and Range

for both single- and double-precision IEEE floating point numbers:

Property	Value for float	Value for double
Largest representable number	3.402823466e+38	1.7976931348623157e+308
Smallest number without losing precision	1.175494351e-38	2.2250738585072014e-308
Smallest representable number(*)	1.401298464e-45	5e-324
Mantissa bits	23	52
Exponent bits	8	11
Epsilon(**)	1.1929093e-7	2.220446049250313e-16

http://www.cprogramming.com/tutorial/floating_point/understanding_floating_point_representation.html

What about Python ? Let’s go back to the Notebook