Computing lecture 3

More data analysis tools

I. PAW macros

the PAWN macros

II. n-tuples

in FORTRAN

in PAW

III. Using CERN library routines

numerical methods, random numbers, etc.

CERNLIB with C++

The PAWN macros

- Many operations in PAW can be painfully tedious

  → PAWN macros by Gerry Lynch (LBL), see course web site

- Need to download `pawn.f`, `pawn.kumac`, `pawn_alias.kumac`, and do `exec pawn_alias` in e.g. `.pawlogon.kumac`

- Based on macros that call HBOOK subroutines

- The basic commands on e.g. histogram 17 (type `pawn` to see all):

`pdef 17`	show booking info about histogram 17
`chbn 17 4`	reduce number of bins by a factor (e.g. by 4); new id chosen and histogram created automatically
`chti 17`	prompts for new title for histogram 17
`mami 17 0 100`	sets min/max of vertical axis (e.g. to 0, 100)
`som 17`	integrates histogram
`blow 17 x1 x2`	blows up histogram to be between `x1` and `x2`
`scal 17 s`	scale histogram by factor `s`

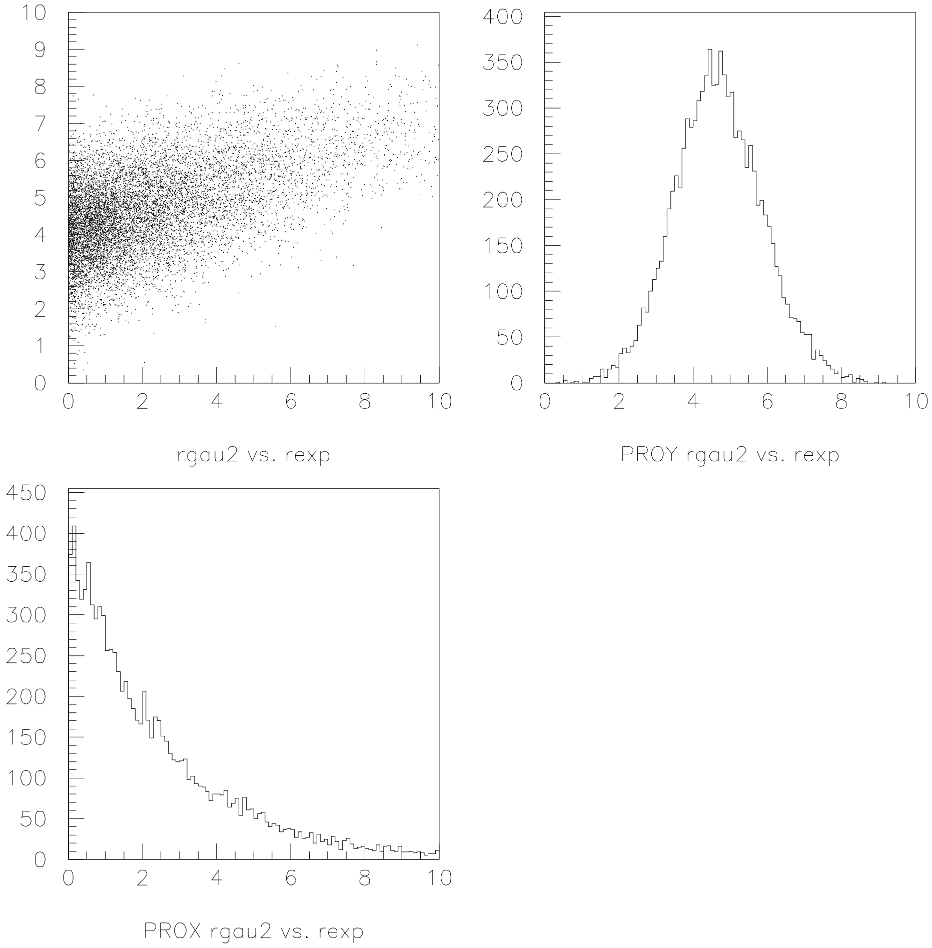
- More commands for operations on scatter plots:

  `chbn2, blo2, prox, proy, regx, regy, avx, avy`

  N.B. these work on 2-d histograms created with HBOOK2, not on scatter plots displayed with e.g. `nt/pl y%x`

Projections from scatter plots with PAWN macros

`zone 2 2`	$\leftarrow$ makes 2×2 windows
`h/pl 1`	$\leftarrow$ show the scatter plot, id=1
`proy 1`	$\leftarrow$ make projection onto y axis
`NEW HISTOGRAM ID 10001`	$\leftarrow$ appears on screen
`h/pl 10001`	$\leftarrow$ plot histogram as usual
`prox 1`	$\leftarrow$ make projection onto x axis
`NEW HISTOGRAM ID 10002`	$\leftarrow$ appears on screen
`h/pl 10002`	



n-tuples

- An n-tuple is a matrix of numbers, e.g. m instances of an n-dimensional vector:

$$
m \text{ `events' (rows)} \left\{ \begin{array}{l}
(x_1, x_2, \quad \ldots \quad , x_n)_1 \\
(x_1, x_2, \quad \ldots \quad , x_n)_2 \\
(x_1, x_2, \quad \ldots \quad , x_n)_3 \\
\qquad\qquad \vdots \\
\underbrace{(x_1, x_2, \quad \ldots \quad , x_n)}_{n \text{ columns, i.e. } n \text{ variables for each event}}{}_m
\end{array} \right.
$$

- Example: for 3-body decays of a certain type of particle, record:

$$p_{x_1}, p_{y_1}, p_{z_1}, p_{x_2}, p_{y_2}, p_{z_2}, p_{x_3}, p_{y_3}, p_{z_3}$$

- Data volume = number of events $\times$ number of columns

- Compare to histogram with n dimensions: data volume = N_{bins}^n

  $\Rightarrow$ for large enough n (usually 2 or more), n-tuple wins

- Use n-tuple to store event properties for use in further analysis, e.g. to make histogram of invariant mass of two of the particles

$$m_{12} = \left[(E_1 + E_2)^2 - (\vec{p}_1 + \vec{p}_2)^2 \right]^{1/2}$$

after making a cut on m_{23} .

n-tuples with HBOOK

- HBOOK provides two types of n-tuples:

  (1) Row wise:

  each event (row) stored sequentially;

  events have fixed length;

  only real (floating point) variables allowed;

  to book: `HBOOKN`

  to fill: `HFN`

  to read: `HGN, HGNPAR, HGNF` (or use PAW)

  (2) Column wise:

  columns stored sequentially

  → faster access if only a few columns out of many needed

  events can contain mixed variable types;

  events can contain data structures of variable length;

  to book: `HBNT, HBNAME`

  to fill: `HFNT, HFNTB`

  to read: `HGNT, HGNTB, HGNTV, HGNTF` (or use PAW)

- See examples on course web site, PAW/HBOOK documentation.

n-tuples with PAW

- The usual procedure:

  create n-tuple in analysis program, then analyze it with PAW

- Suppose we've created an n-tuple with 3 columns: x, y, z.

  Read in with **`h/file`** as before; suppose n-tuple id is 17.

  The basic commands:

  `ntuple/plot 17.x` ← plot histogram of x

  `nt/pl 17.y%x` ← make scatter plot of y vs. x

  `nt/print 17` ← show variable names, min/max values

  `nt/scan 17` ← show entire contents of n-tuple

  `nt/pl 17.x 2*y-z<1.5` ← histogram x after cut on y, z

  `nt/cut $1 2*y-z<1.5` ← define cut number 1

  `nt/pl 1.x $1` ← histogram of x after cut 1

- To set histogram properties (e.g. id=20, 100 bins from 0 to 50):

  `1dhisto 20 'histo of x' 100 0. 50.`

  `nt/proj 20 17.x` ← project contents into histogram

  `h/pl 20` ← display histogram as usual

- More advanced features possible (loops, masks, . . .) but at some point it's better to use a high level program (FORTRAN, C++).

The CERN program library (CERNLIB)

- Web site: `http://wwwinfo.cern.ch/asd/index.html`
- CERNLIB contains FORTRAN subprograms for:

  numerical analysis

  Monte Carlo

  data handling (histograms, search, sort, . . .)

  event generators, detector simulation, particle kinematics, . . .
- LHC++ project: replace CERNLIB with OO (e.g. C++) libraries

  only partially complete, rapid state of flux
- Provisional solution for C++:

  call CERNLIB routines from C++ using `cfortran.h`, see

  `http://wwwinfo.cern.ch/asdcgi/listpawfaqs.pl/21`
- To use, include at end of link command:

  `‘cernlib‘` or if more libraries needed, try e.g.

  `‘cernlib graflib mathlib kernlib packlib‘`
- Some other libraries (see links on course web site):

  Statlib (Carnegie Mellon)

  Netlib (University of Tennessee)

  Numerical Algorithms Group (NAG)

  Numerical Recipes (Press et al.)

  Datenanalyse (Brandt)

Calling CERNLIB routines from C++ programs

- For details see

  http://wwwinfo.cern.ch/asdcgi/listcernlibfaqs.pl/10

  and example `test_radom.cc` on SDA web site.

- Need C++ header files in `/cern/pro/include/cfortran`

- To call a FORTRAN CERNLIB routine from C++,

  (1) Include the necessary header files:

```
#include "cfortran/cfortran.h"
#include "cfortran/hbook.h"
```

  (2) Call routine (name in capitals) with same parameters:

```
main(){
                    ⋮
   HBOOK (1, "test histogram", 100, 0., 1., 0.);
   for (int i = 1; i<1000; i++){
      float x = random(seed);
      HF1 (1, x, 1.);
   }
                    ⋮
}
```

- Special flags needed to compile/link (see documentation on web)