

Histograms, etc.

I. In general

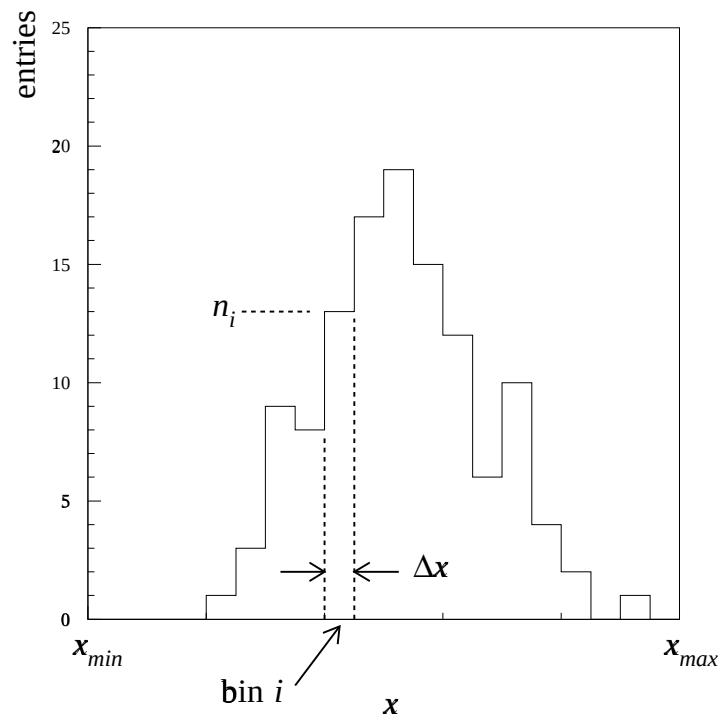
II. In FORTRAN

III. In PAW

Histograms

- Consider a data sample $\vec{x} = (x_1, \dots, x_m)$ (m can be large)

→ summarize information
as a histogram
(N bins)



- Generic computer implementation:
 - Define bins, e.g. N bins of width Δx from x_{min} to x_{max} .
 - Declare variables to hold $n_1, \dots, n_N$, initialize all n_i to 0.
 - Loop over $x_1, \dots, x_m$; if x value in bin i , $n_i \rightarrow n_i + 1$.
- In practice, not trivial $\Rightarrow$ use packages HBOOK, HTL, ...

- The HBOOK package (from CERN): user-callable FORTRAN subroutines for creating/manipulating:

histograms (1-dimensional)
scatter-plots (2-dimensional)
 n -tuples

- The basic steps to get a histogram:

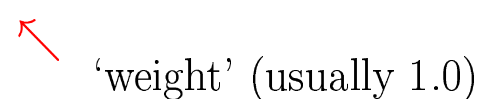
I. ‘Book’ histogram: define bins, allocate memory for $n_1, \dots, n_N$.

```
call HBOOK1(17, 'x values', 100, xmin, xmax, 1.)
```


id number title number of bins

II. ‘Fill’ the histogram:

```
do i = 1, m
  call HF1(17, x(i), 1.)
end do
```

 ‘weight’ (usually 1.0)

- Steps also needed to set up output file and store results
(see example on next page)

```
      program TEST_HBOOK

c  Glen Cowan
c  5 October, 1999
c  Test program for using HBOOK

      implicit      NONE

c  Needed for HBOOK routines

      integer      hsize
      parameter    (hsize = 100000)
      integer      hmemor (hsize)
      common /pawc/ hmemor

c  Local variables

      character*80  outfile
      integer       i, icycle, istat, num_values
      real          x

c  Initialize HBOOK, open histogram file, book histograms.

      call HLIMIT (hsize)
      outfile = 'test_hbook.his'
      call HROPEN (20, 'histog', outfile, 'N', 1024, istat)
      call HBOOK1 (17, 'x values', 100, 0., 10., 0.)

c  Get x values and enter into histogram.

      write (*, *) 'enter number of x values to get'
      read (*, *) num_values
      do i = 1, num_values
         call GET_ANOTHER_X_VALUE (x)      ! subroutine supplied by user...
         call HF1 (17, x, 1.)
      end do

c  Store histogram and close.

      call HROUT (0, icycle, ' ')
      call HREND ('histog')

      stop
      END
```

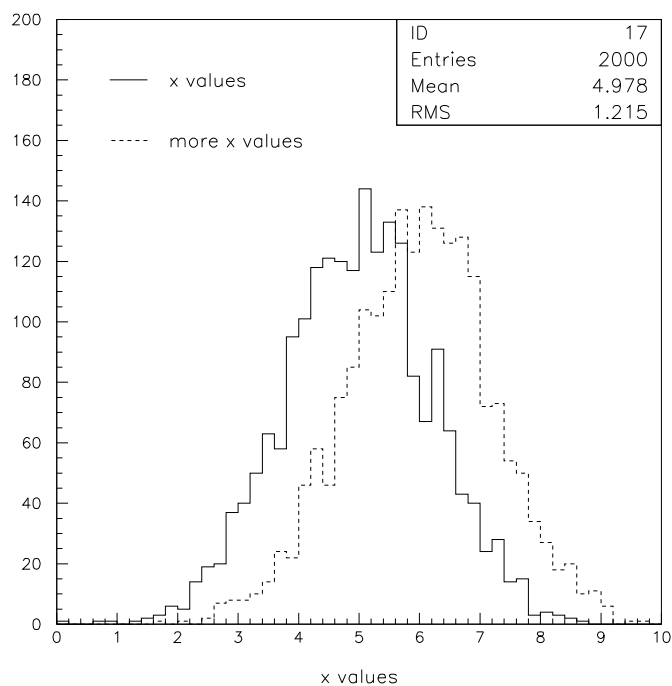
Looking at the histograms with PAW

- Running the sample program creates the file `test_hbook.his`.
To view/manipulate the histograms with PAW,

```
h/file 1 test_hbook.his    ← read in file
h/list                     ← show list of histograms

==> Directory :
17 (1) x values
23 (1) more x values

h/pl 17                    ← plot histogram 17
h/pl 23 s                  ← put 23 on same plot
```



- See documentation for details on commands like:

```
opt stat, set dmod, h/set max, key, ...
```

Two-dimensional histograms (scatter plots)

- Bins are now cells in 2-d plane. HBOOK routines similar to 1-d:

I. To book:

```
call HBOOK2 (37, 'y vs. x', nx, xmin, xmax,  
& ny, ymin, ymax, 0.)
```

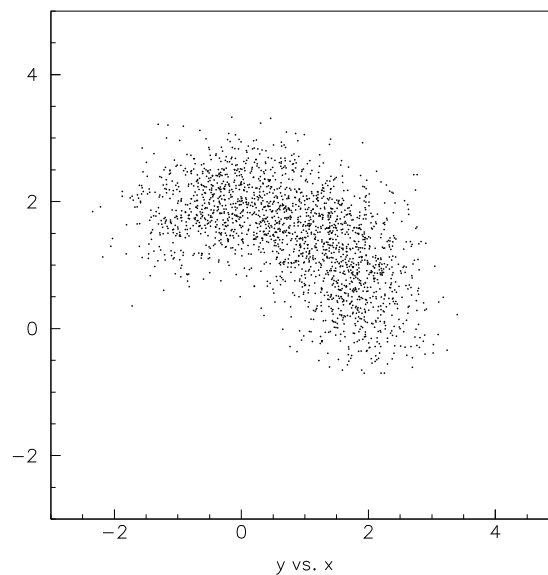
Annotations:

- id number (points to 37)
- title (points to 'y vs. x')
- number of bins in x (points to nx)
- same stuff for y (points to ny, ymin, ymax)

II. To fill:

```
call HF2 (37, x, y, 1.)
```

- Viewing with PAW same as in 1-d case:



N.B. Exact x, y values not recorded, only numbers of entries in each bin ($N_x \times N_y$ values stored).

Access to information

- In FORTRAN (see HBOOK manual for details)

Open file and read in the histograms:

```
call HROPEN (30, ' ', 'myfile.his', ' ', lrec, istat)
call HRIN (0, icycle, 0)
```

Access contents of histograms, errors, etc.

```
call HNOENT (id, num_entries)      ← number of entries
call HUNPAK (id, contents, ' ', 0) ← unpack into array
call HGIVE (id, title, nx, xmin, xmax, ny,
& ymin, ymax, nwt, loc)           ← get booking info
```

- In PAW (see PAW manual or online help)

Read histograms into memory, use variables and system functions,
best used in macros ('kumac' files).

```
h/file 1 myfile.his
hrin 0      ← read histograms into memory
id = 17     ← define variable id, use brackets to evaluate.
nx = $HINFO([id], 'XBINS')  ← system function HINFO
vec/create myvector([nx]) R ← vector to hold histogram
vec/print myvector          ← show vector contents
mess 'events =' $HINFO([id], 'EVENTS') ← # entries
```