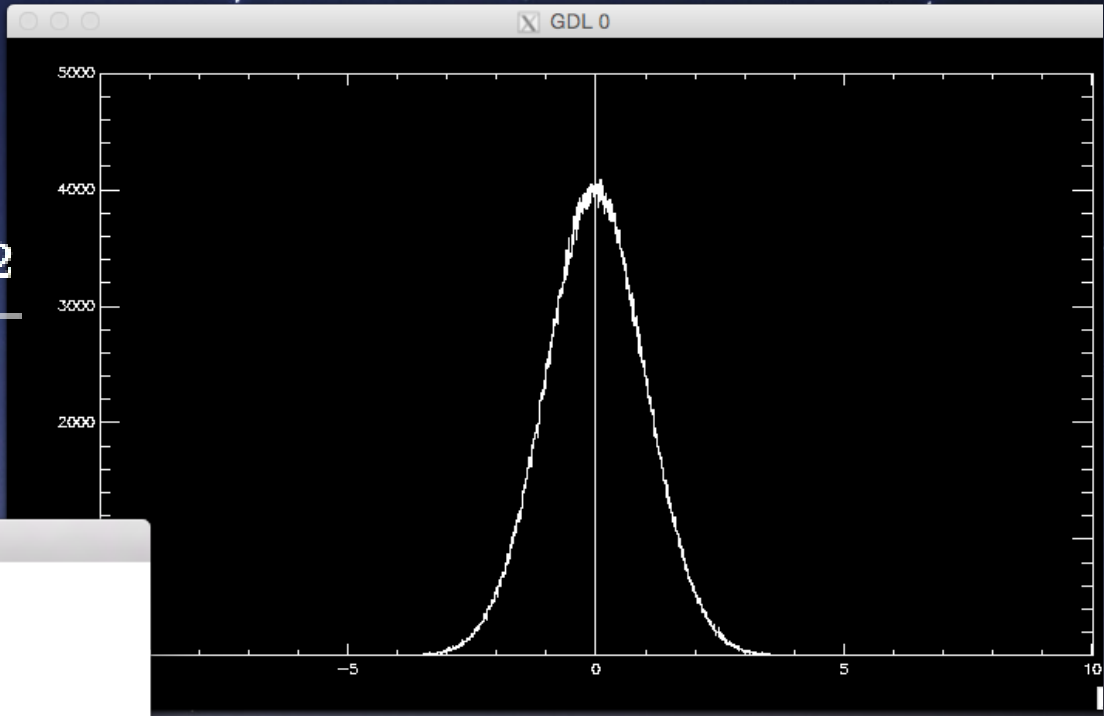
Statistics with Normal Distributions

Jayant Murthy
The Indian Institute of Astrophysics
www.iiap.res.in
jmurthy@yahoo.com

Normal Distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



1. gdl

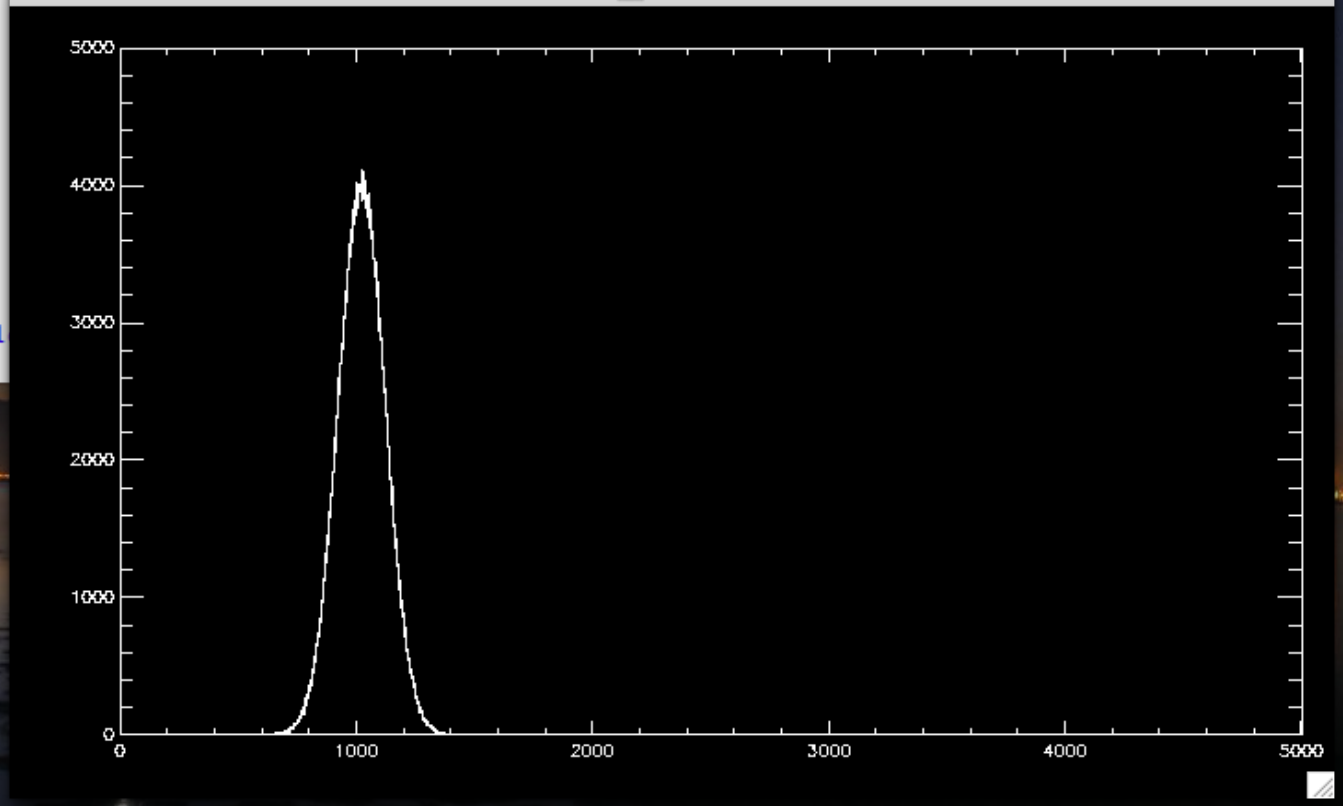
```
  GDL - GNU Data Language, Version 0.9.5

- For basic information type HELP,/INFO
- No startup file read (GDL_STARTUP/IDL_STARTUP env. var. not set).
- Please report bugs, feature or help requests and patches at:
  http://sourceforge.net/projects/gnudatalanguage/

GDL> n=randomn(seed,1000000)
GDL> print,mean(n),stdev(n)
% Compiled module: MEAN.
% Compiled module: STDEV.
  0.000288876      1.00057
GDL> h=histogram(n,min=-10,bin=.01,max=10)
GDL> help,h
H               LONG      = Array[2001]
GDL> plot,findgen(2001)*.01-10,h,psym=10

*** PLPLOT WARNING ***
You said you want pthreads, but they are not available.
GDL> oplot,[0,0,],[5000,5000]
% Parser syntax error: unexpected token: ]
GDL> oplot,[0,0],[5000,5000]
GDL> oplot,[0,0],[0,5000]
GDL>
```

But I don't want a mean of 0

1. gdl

```
GDL> oplot,[0,0,],[5000,5000]
% Parser syntax error: unexpected token: ]
GDL> oplot,[0,0],[5000,5000]
GDL> oplot,[0,0],[0,5000]
GDL>
$gdl

  GDL - GNU Data Language, Version 0.9.5

- For basic information type HELP,/INFO
- No startup file read (GDL_STARTUP/IDL_STARTUP env. var. not set).
- Please report bugs, feature or help requests and pa
  http://sourceforge.net/projects/gnudatalanguage/

GDL> n=randomn(seed,1000000)*100+1024
GDL> print,mean(n),stdev(n)
% Compiled module: MEAN.
% Compiled module: STDEV.
      1023.96      99.9701
GDL> h=histogram(n,min=0,bin=1,max=5000)
GDL> plot,findgen(5000),h,psym=10

*** PLPLOT WARNING ***
You said you want pthreads, but they are not availabl
GDL>
```

GDL 0



Definitions

- $\mu = (\Sigma\, y_i)/N$
- $\sigma^2 = \Sigma(y_i - \mu)^2/(N - 1)$
- mode = mean

1. gdl

```
- For basic information type HELP,/INFO
- No startup file read (GDL_STARTUP/IDL_STARTUP env. var. not set).
- Please report bugs, feature or help requests and patches at:
  http://sourceforge.net/projects/gnudatalanguage/

GDL> n=randomn(seed,1000000)
GDL> print,mean(n),total(n)/1000000.
% Compiled module: MEAN.
  0.000497694  0.000497694
GDL> m=mean(n)
GDL> print,stdev(n),sqrt(total((n-m)^2)/1000000)
% Compiled module: STDEV.
     0.999429     0.999429
GDL> s=sort(n)
GDL> print,s(500000)
      170360
GDL> print,median(n),n(s(500000))
  0.000258923  0.000258923
GDL>  h=histogram(n,min=-10,bin=.01,max=10)
GDL> print,where(h eq max(h))
         995
GDL> print,where(h eq max(h))*.01+(-10)
   -0.0500002
GDL>
```

FWHM

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- $\exp(-x^2/2\sigma^2) = 0.5$
- $x^2/2\sigma^2 = \ln(2)$
- $x = \mathrm{sqrt}(2\ln(2))\sigma$
- FWHM = 2x
- FWHM ~ 2.355σ

1. vim

```
nmax = 1000000
for i = 0,10 do begin
n = randomn(seed, nmax)
h = histogram(n, min = -10, bin = 0.001, max = 10)
hmax = where(h eq max(h))
hval = h(hmax(0))
fmin = min(where(h gt (hval/2.)))
fmax = max(where(h gt (hval/2.)))
print,(fmax - fmin)*.001
endfor
end
```

2. gdl

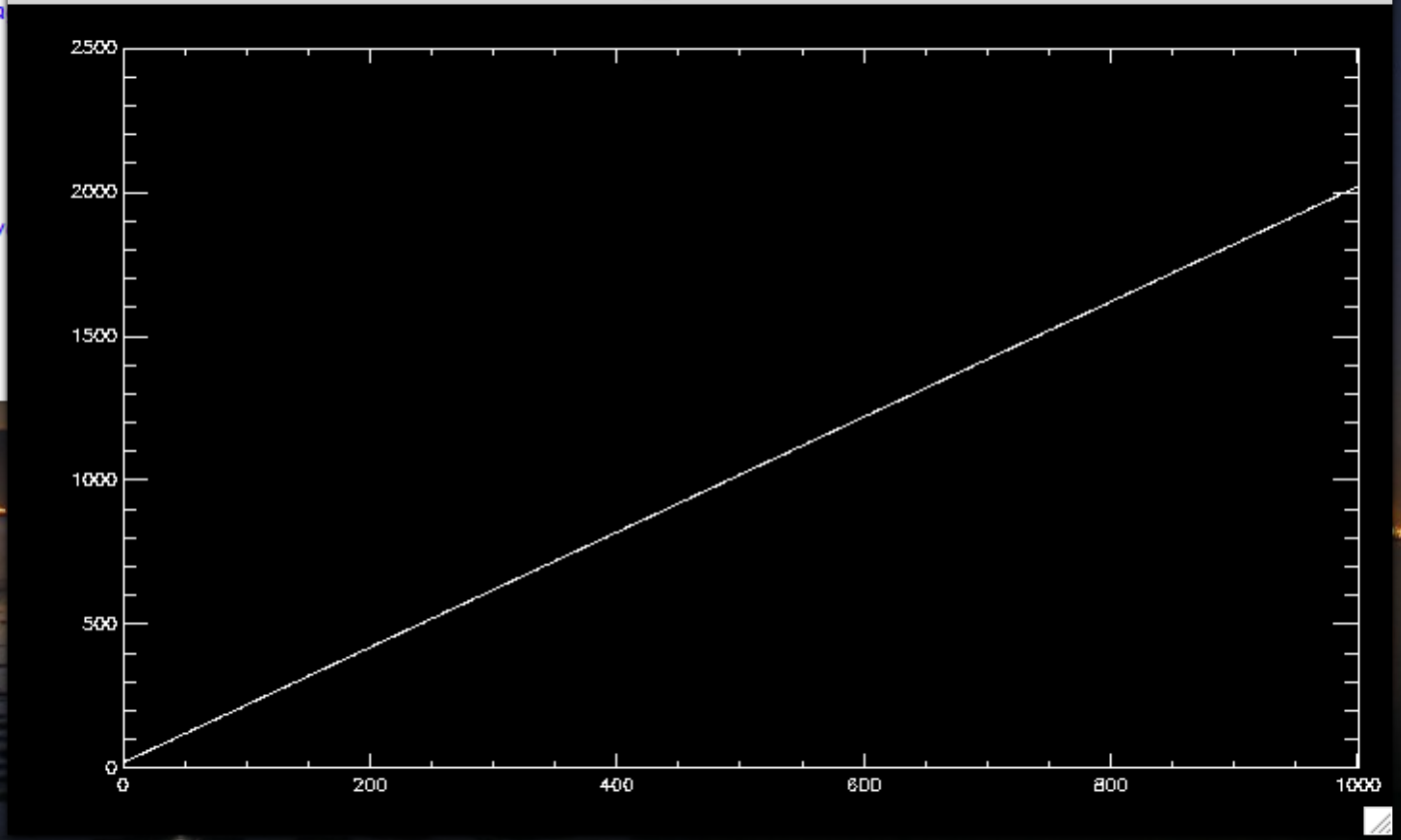
```
      2.42000
GDL> .run find_normal_fwhm.pro
% Compiled module: $MAIN$.
       10025
         451
      2.37800
GDL> .run find_normal_fwhm.pro
% Compiled module: $MAIN$.
        9793
         455
      2.35200
GDL> .run find_normal_fwhm.pro
% Compiled module: $MAIN$.
      2.26900
      2.34800
      2.26800
      2.35900
      2.34100
      2.32500
      2.34400
      2.37300
      2.31100
      2.35300
      2.36700
```

Fitting Data

2. gdl

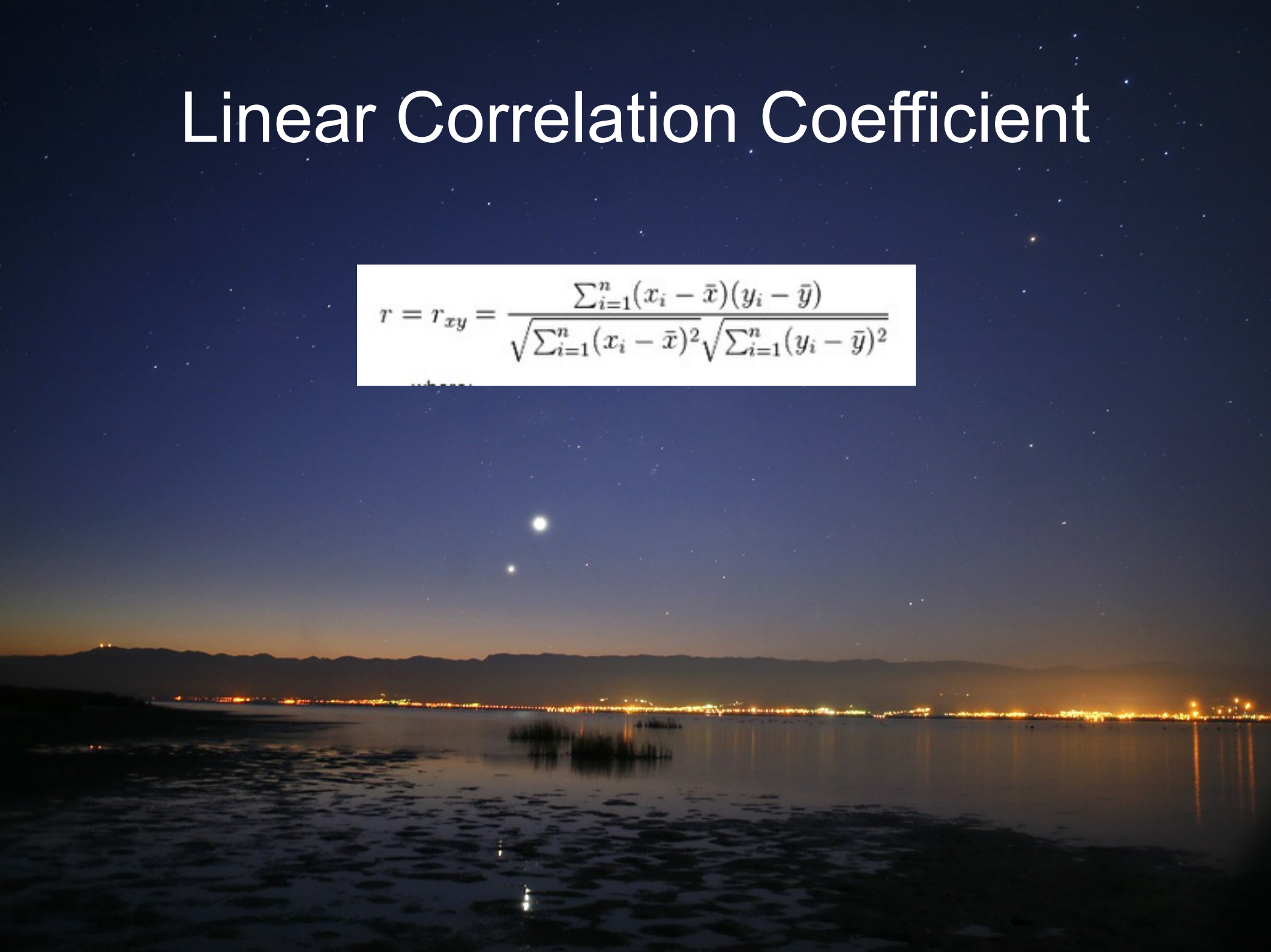
```
GDL> print,correlate(x,y)
% Compiled module: CORRELATE.
% Compiled module: MEAN.
     1.000000
GDL>
$gdl

  GDL - GNU Data Language, Version 0.9.5

- For basic information type HELP,/INFO
- No startup file read (GDL_STARTUP/IDL_STARTUP env. var. not set).
- Please report bugs, feature or help requests
  http://sourceforge.net/projects/gnudatalangua

GDL> x=findgen(1000)
GDL> y=2*x+20
GDL> plot,x,y

*** PLPLOT WARNING ***
You said you want pthreads, but they are not av
GDL> print,correlate(x,y)
% Compiled module: CORRELATE.
% Compiled module: MEAN.
     1.000000
GDL>
```

GDL 0



Linear Correlation Coefficient

$$r = r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$



$$r = r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

```
; by Sylwester Arabas <slayoo@igf.fuw.edu.pl>
; part of GNU Data Language - GDL
function correlate, x, y, covariance=covariance, double=double
  on_error, 2

  if n_params() eq 2 then begin

    l = n_elements(x) < n_elements(y)
    mx = mean(x[0:l-1], double=double)
    my = mean(y[0:l-1], double=double)
    cov = total((x - mx) * (y - my)) / (l - 1.)
    if keyword_set(covariance) then return, cov

    sx = sqrt(total((x[0:l-1] - mx)^2, double=double) / (l - 1.))
    sy = sqrt(total((y[0:l-1] - my)^2, double=double) / (l - 1.))
    return, cov / sx / sy

  endif else if n_params() eq 1 then begin
```

returns control to calling program

there have to be 2 parameters

pick the lesser of number of elements of x and y

find mean of x and y. If double is set calculations are done in double precision.

Calculate the numerator.

Return the covariance matrix if keyword set.

Calculate correlation coefficient.

Return correlation coefficient

Aside: Note on Vectors in IDL

2. gdl

```
GDL> a = 5 & b = 6 & print,a*b
      30
GDL> a = findgen(5) & b = 6 & print,a*b
      0.00000      6.00000      12.0000      18.0000      24.0000
GDL> a = findgen(5) & b = findgen(5) & print,a*b
      0.00000      1.00000      4.00000      9.00000      16.0000
GDL> print,a#b
      0.00000      0.00000      0.00000      0.00000      0.00000
      0.00000      1.00000      2.00000      3.00000      4.00000
      0.00000      2.00000      4.00000      6.00000      8.00000
      0.00000      3.00000      6.00000      9.00000      12.0000
      0.00000      4.00000      8.00000      12.0000
GDL> for i = 0,4 do for j=0,4 do print,i,j,a(i)*b(j)
       0       0      0.00000
       0       1      0.00000
       0       2      0.00000
       0       3      0.00000
       0       4      0.00000
       1       0      0.00000
       1       1      1.00000
       1       2      2.00000
       1       3      3.00000
       1       4      4.00000
       2       0      0.00000
       2       1      2.00000
```

2. gdl

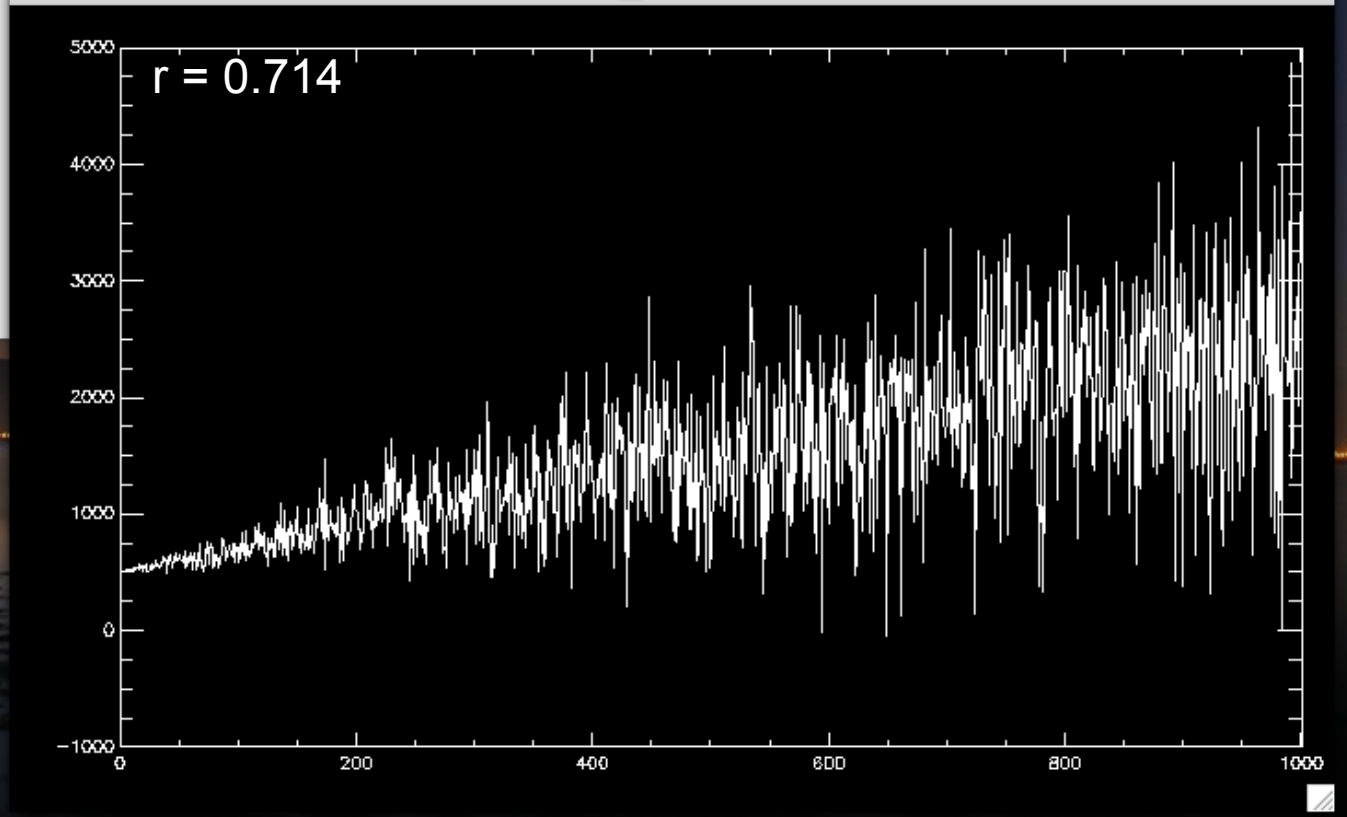
```
GDL> t=systime(1) & for i=0l,10000 do for j=0l,10000 do c=i*j & print,systime(1)
 - t
       14.502406
GDL> t=systime(1) & c=lindgen(10001)*lindgen(10001) & print,systime(1)-t
    0.0076589584
GDL>
$gdl

  GDL - GNU Data Language, Version 0.9.5

- For basic information type HELP,/INFO
- No startup file read (GDL_STARTUP/IDL_STARTUP env. var. not set).
- Please report bugs, feature or help requests and patches at:
  http://sourceforge.net/projects/gnudatalanguage/

GDL> t=systime(0) & print,t
Mon Jul 13 09:32:24 2015
GDL> t=systime(1) & print,t
   1.4367601e+09
GDL> t=systime(1) & for i=0l,10000 do for j=0l,10000 do c=i*j & print,systime(1)
 - t
       14.309439
GDL> t=systime(1) & c=lindgen(10001)*lindgen(10001) & print,systime(1)-t
   0.00010514259
GDL>
```

Adding Noise

2. vim

```
nelems = 1000
x = findgen(nelems)
y = x*2. + 500.
plot,x,y

;add noise
y = (2. + randomn(seed,nelems))*x + (500. + randomn(seed,nelems))
plot,x,y

end
~
~
~
~
~
~
~
~
~
~
~
~
~
"add_noise.pro" [New] 10L, 151C written
```

GDL 0



Calculating Best Fit Parameters

- Metric: least squares fit.
  - $y = ax + b$
  - $LS = \Sigma(y_i - (ax_i + b))^2$ where we solve for a and b
- Best fit for a and b will be when
  - $\delta LS/\delta a = 0$
  - $\delta LS/\delta b = 0$

$$\frac{\partial \sum (y_i - ax_i - b)^2}{\partial a} = 0$$

$$\sum (y_i - ax_i - b)x_i = 0 \Rightarrow$$

$$\sum y_i x_i - a\sum x_i^2 - b\sum x_i = 0$$

$$\& \quad \sum (y_i - ax_i - b) = 0$$

$$\sum y_i - a\sum x_i - Nb = 0$$

$$\boxed{b = \frac{\sum y_i - a\sum x_i}{N}}$$

$$\sum y_i x_i - a\sum x_i^2 - \frac{\sum y_i - a\sum x_i}{N}\sum x_i = 0$$

$$\sum y_i x_i - \frac{1}{N}\sum y \sum x_i = a\left(\sum x_i^2 - \left(\sum x_i\right)^2/N\right)$$

$$\boxed{a = \frac{\sum y_i x_i - \frac{1}{N}\sum y_i \sum x_i}{\sum x_i^2 - \left(\sum x_i\right)^2/N}}$$

Note that the linearity is for a and b.

Analytical Programming

2. vim

```
nelems = 1000
x = findgen(nelems)

for i=0,10 do begin
y = (2. + randomn(seed,nelems))*x + (500. + randomn(seed,nelems))
plot,x,y

yx = total(x*y)
sumyx = total(x)*total(y)/float(nelems)
xsqr = total(x*x)
sum_x_sqr = total(x)*total(x)/float(nelems)
a = (yx - sumyx)/(xsqr - sum_x_sqr)

b = (total(y) - total(x)*a)/float(nelems)
print,a,b
endfor
end
~
~
~
~
~
~
~
"analytical_least_squares.pro" 17L, 349C written
```

1. gdl

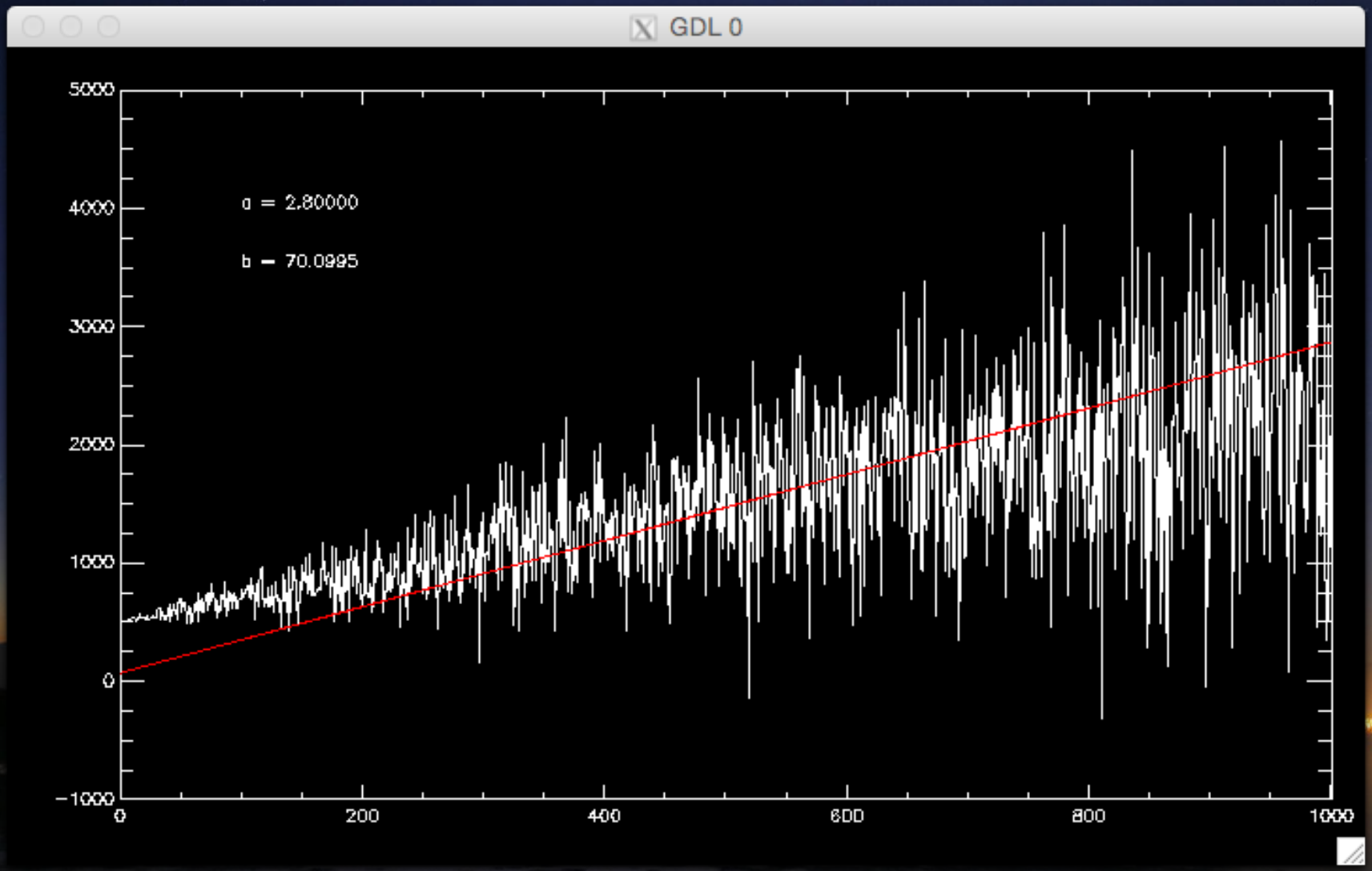
```
*** PLPLOT WARNING ***
You said you want pthreads, but they are not available.
GDL> .run analytical_least_squares.pro
% Compiled module: $MAIN$.
      1.95728      524.696
GDL> .run analytical_least_squares.pro
% Compiled module: $MAIN$.
      1.96970      520.578
GDL> .run analytical_least_squares.pro
% Compiled module: $MAIN$.
      2.06189      464.039
GDL> .run analytical_least_squares.pro
% Compiled module: $MAIN$.
      2.06040      476.865
      2.02218      476.924
      1.84081      529.854
      1.97541      516.642
      1.96017      520.922
      2.01773      496.797
      1.98126      513.750
      2.05770      499.454
      2.04835      473.004
      2.00549      508.087
      2.09916      488.654
GDL>
```

Brute Force

```
nelems = 1000
x = findgen(nelems)
y = (2. + randomn(seed,nelems))*x +$
 (500. + randomn(seed,nelems))
plot,x,y

a = 0.
b = 0.
dela = .1
delb = .1
model = a*x + b

ls = total((model - y)^2)
ls_old = ls + 1000.

while (ls lt ls_old) do begin
   a = a + dela
   model = a*x + b
   ls = total((model - y)^2)
   if (ls lt ls_old)then begin
        ls_old = ls
        ls = 0
   endif
endwhile
```

```
s = total((model - y)^2)
ls_old = ls + 1000.

while (ls lt ls_old) do begin
   b = b + delb
   model = a*x + b
   ls = total((model - y)^2)
   if (ls lt ls_old)then begin
        ls_old = ls
        ls = 0
   endif
endwhile
print,a,b
oplot,x,model,col=255
end
```

Brute Force Minimization

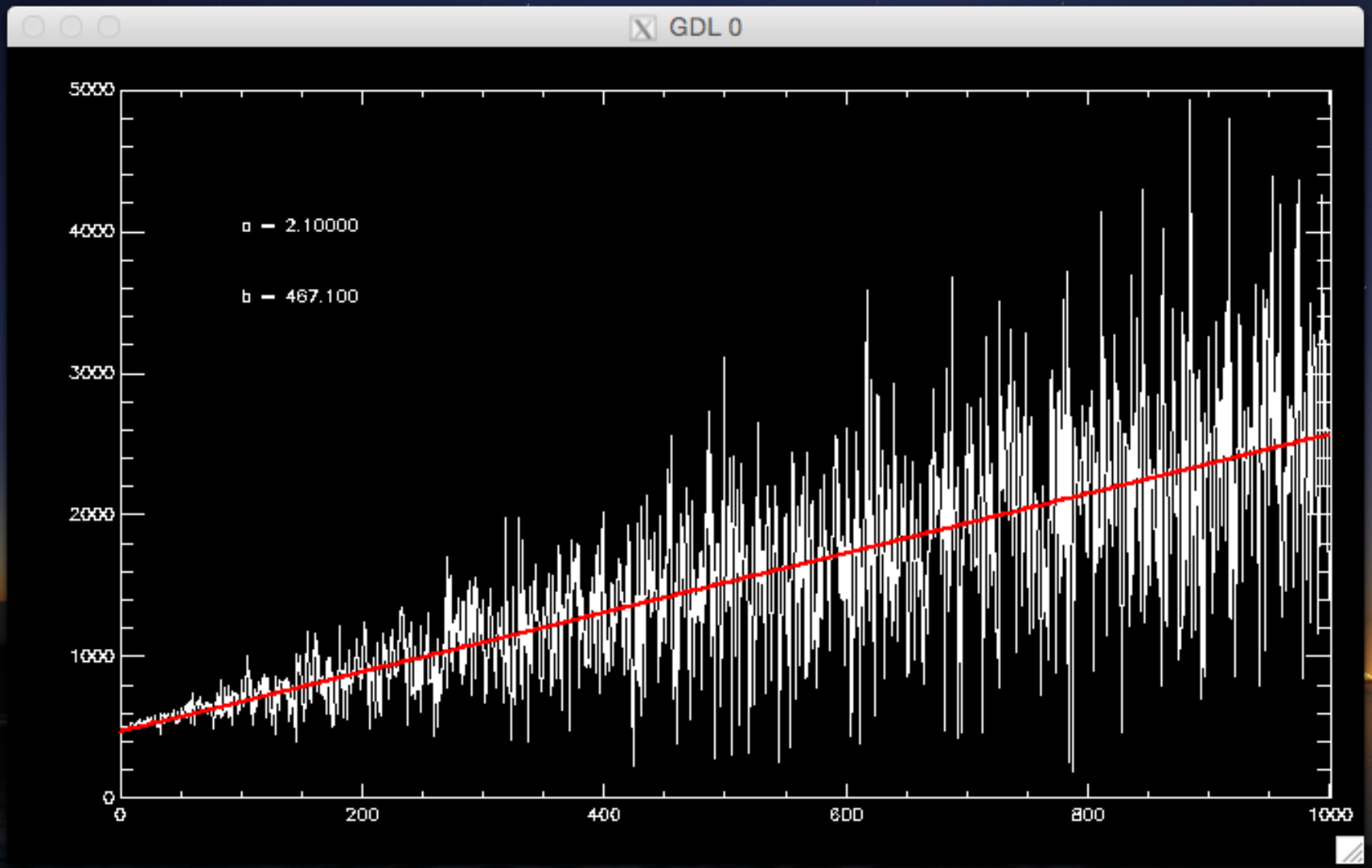



```
xyouts,100,4000,"a = " + strcompress(string(a),/rem)
xyouts,100,3500,"b = " + strcompress(string(b), /rem)
```

Grid Search

```
nelems = 1000
x = findgen(nelems)
y = (2. + randomn(seed,nelems))*x +$
      (500. + randomn(seed,nelems))
plot,x,y
dela = .1
delb = .1
ls = fltarr(100,10000)
a = lindgen(100,10000) mod 100
a = float(a)/10.
b = lindgen(100,10000)/100
b = float(b)/10.

t0 = systime(1)
for i = 0,99 do begin
print,i,(systime(1) - t0)/float(i)*(100. -i)
  for j = 0,9999 do begin
    model = a(i,j)*x + b(i,j)
    ls(i,j) = total((model - y)^2)
  endfor
endfor

q=where(ls eq min(ls)) & q = q(0)
print,a(q),b(q),ls(q)
model = a(q)*x + b(q)
oplot,x,model,col=255,thick =2
xyouts,100,4000,"a = " + strcompress(string(a(q)),/rem)
xyouts,100,3500,"b = " + strcompress(string(b(q)), /rem)
end
```

Grid Search


GDL 0
a = 2.10000
b = 467.100
5000
4000
3000
2000
1000
0
0
200
400
600
800
1000

Importance of σ

- Results are often quoted as ±1σ.
- 3σ or 5σ are *detections.*
- Numbers to remember:
  - 68% within 1σ of mean.
  - 95% within 2σ of mean.
  - 99% within 3σ of mean.

Sigma for Normal Distributions

1. gdl

```
- No startup file read (GDL_STARTUP/IDL_STARTUP env. var. not set).
- Please report bugs, feature or help requests and patches at:
  http://sourceforge.net/projects/gnudatalanguage/

GDL> n=randomn(seed,1000000)
GDL> h=histogram(n,min=-10,bin=.01,max=10)
GDL> plot,findgen(2001)*.01-10,h,psym=10

*** PLPLOT WARNING ***
You said you want pthreads, but they are not available.
GDL> print,mean(n),stdev(n)
% Compiled module: MEAN.
% Compiled module: STDEV.
  0.000425670     0.999279
GDL> print,total(h(900:1100))/total(h)
     0.685327
GDL> print,total(h(800:1200))/total(h)
     0.955208
GDL> print,total(h(700:1300))/total(h)
     0.997376
GDL> print,total(h(500:1500))/total(h)
     0.999998
GDL> print,(total(h(0:499)) + total(h(1501:*)))/total(h)
  2.00000e-06
GDL>
```

GDL 0