

# Some APL Examples

By Jerry Brennan  
Jerrymbrennan.com  
jbrennan@hawaii.rr.com  
538-0343

## Table of Contents

The Birthday problem .....	2
Now Lets Plot These Probabilities .....	3
Two Dice - How Lucky Are You?.....	5
Probability of two dice being equal.....	6
The Power of 11.....	11
Mortgage Calculations.....	14
Roots of a Polynomial.....	15
Quadric Equations and Functions.....	20
Basic Statistics: .....	21
Linear regression: compute line from raw data.....	23
Linear Quadratic and Cubic Regression.....	26
Plotting 3 Exponential Functions to compare.....	27
Plotting in General in APL.....	28
Are all Numbers of Form abcabc Divisible by 13?.....	28
Rate Writing Based Upon Word And Sentence Length.....	29
What Is Your Name Worth?.....	31
Working with Tables.....	33
Plotting Regular Polygons.....	35
APL References.....	37

## THE BIRTHDAY PROBLEM

If you go to a party and there are 35 people there what is the chance that two of the people will have the same birthday.

From Wolfram: They odds are about 81%. The formula is listed below. <http://www.wolframalpha.com/input/?i=birthday+problem+35+people>

Input information:

birthday problem	
number of people	35

Result:

probability at least two with the same birthday	0.8143832388747246
---	--------------------

Equation:

$\text{Pr} = 1 - \frac{365!}{365^n (365-n)!}$	
Pr	probability at least two with the same birthday
n	number of people

n! is the factorial function »

Computed by **Wolfram Mathematica**

Download as: [PDF](#) | [Live Mathematica](#)

In APL you can easily create a program to calculate the formula like this:

```
birthdaysame←{⌊FR←1287 ⋄ 1-(!365)÷(365*ω)×(!365-ω)}
```

The ⌊FR←1287 tells apl to do this in double precision arithmetic which is needed because of the large factorial and power calculations.

The ω stands for n in the above equation i.e. the number of people at the party. In APL the factorial symbol goes in front of the number. Also in APL calculation goes from right to left so the entire denominator is calculated first, then the division occurs and finally the subtraction from 1.

Now lets try out the program for the 35 people at the party.

```
Birthdaysame 35  
0.8143832389
```

So there is about an 81% chance that two people will have the same birthday. Lets try a couple of others and see the percents.

```
birthdaysame 25      A 25 people at the party
0.568699704          A about 57%
birthdaysame" 50 66  A find odds for each("") 50 and 66 people
0.9703735796 0.9980957046  A 97% for 50 and 99.8% for 66 people.
So it looks like once we get to 66 people odds are almost 100%.
```

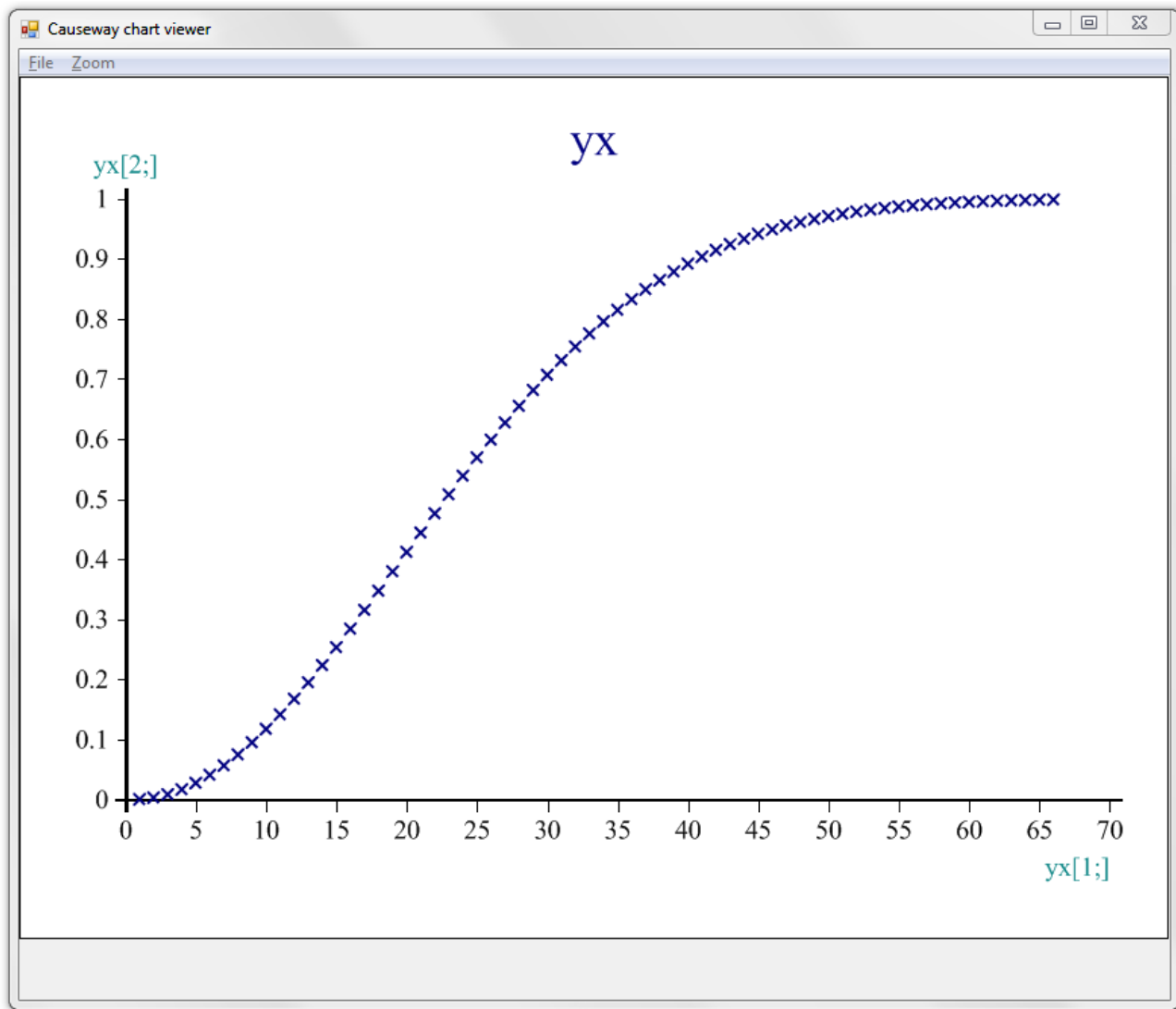
### Now Lets Plot These Probabilities

for all the #'s of people from 1 to 66. Apl has a special operator called iota ( $\iota$ ) that will easily generate all the numbers for one to any number you want.

```

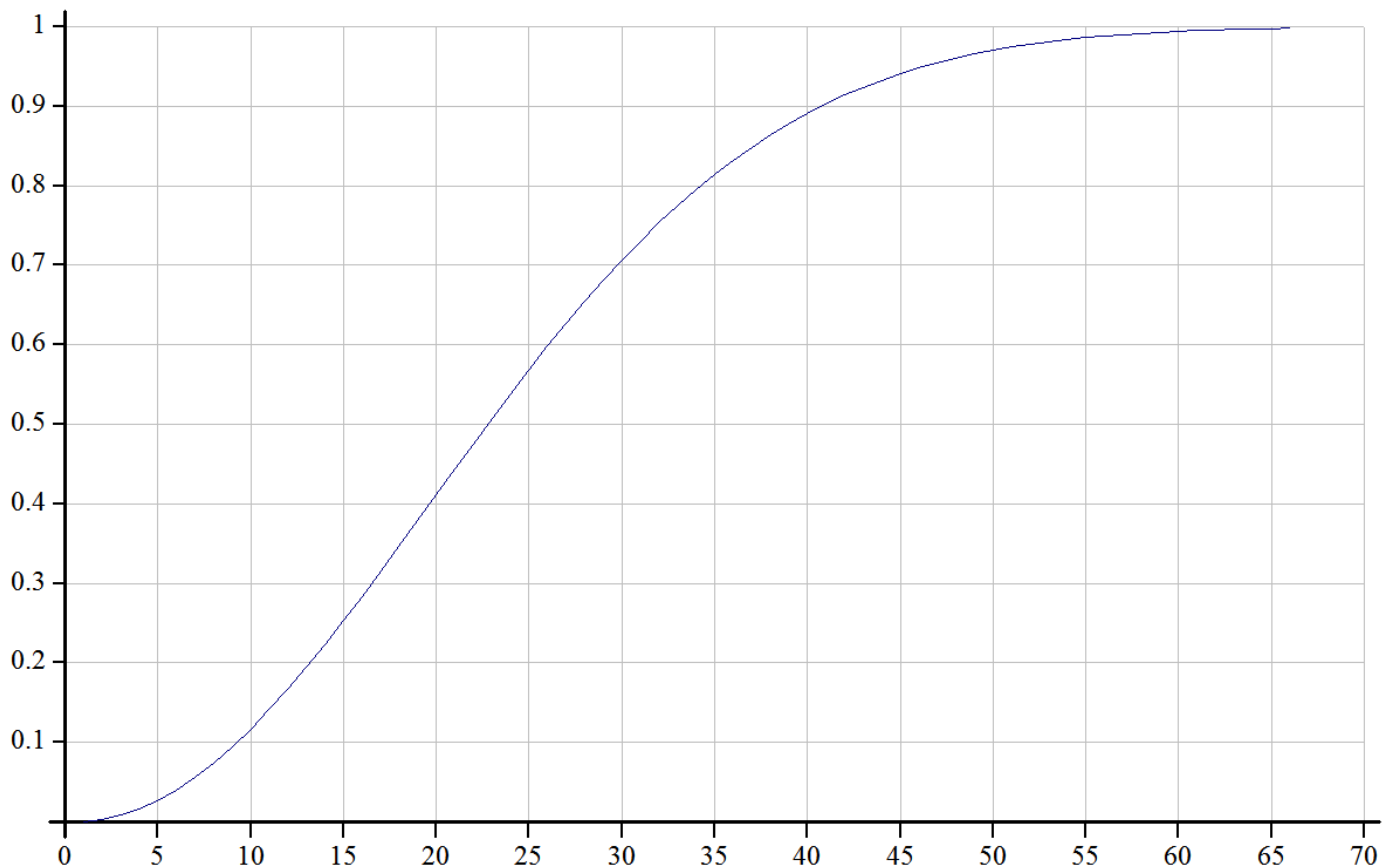
   $\iota$ 6
1 2 3 4 5 6      A monadic  $\iota$  called: index generator makes numbers 1-6
  10+ $\iota$ 8
11 12 13 14 15 16 17 18  A generates numbers 1-8 first then adds 10 to each. So:
```

```
yx← $\phi$ x(y←birthdaysame ""x← $\iota$ 66)  A x=#1-66 y=odds for each("") x. yx is x
and y put together and then reversed( $\phi$ ) to be y and then x
Now put cursor on yx and click the Scatterplot icon at top of screen to see
the plot below: (y axis:the odds for # 1-66 by x axis:the # 1-66) You can
see for example that for 40 people the odds is about 90%.
```



The previous plot is quick and easy but a little hard to read. APL has extremely sophisticated plotting/graphing capabilities and with just a little effort we can produce a prettier plot with grid lines.

```
plotxy x (y←birthdaysame "x←166) A for each # 1 to 66(166) and plot
View PG A to see it           A press enter on this line to see plot
```



Here's the plot fns : To create it type )ed plotxy press enter and type in

```
R←{ax0}plotxy data
```

```
  A plot data:x=col1 y=col2 or x=vector1 y=vector2
```

```
  ax0←0=⎕NC'ax0' A if no ax0 axes cross at 0
```

```
  :If 2=≡data ⋄ data←⌵↑data ⋄ :End
```

```
  ch.Set'Lines' 1 2 4 5
```

```
  ch.Set''(ax0,ax0,1)/('Xint' 0)('Yint' 0)('XYPLOT,GRID')
```

```
  ch.Plot data ⋄ PG←ch.Close
```

```
  R←'View PG A to see it'
```

## TWO DICE - HOW LUCKY ARE YOU?

In APL the ? is used to generate random numbers so

```
  ?6 A generates a random number between 1 and 6 each time you do it
```

```
3      A got a 3 this time
```

```
  ?6
```

```
5      A got a 5 this time
```

To throw two dice you need two 6's

```
  ?6 6
```

```
2 4      A got a 2 and a 4
```

```

dice←{A Here's a program to interpret 2 dice throws. To call: dice ?6 6
      ω≡6 6:ω,'Box Cars'      A if inputs(ω) match(≡)6 6 display Box Cars
      ω≡1 1:ω,'Snake Eyes'    A if inputs(ω) match(≡)1 1 display Snake Eyes
      =/ω:ω,'Pair'            A if inputs(ω) are equal(=/) display Pair
      7=+/ω:ω,'Seven'         A if inputs(ω) sum(+/)=7 display Seven
      ω,'Unlucky'            A else display Unlucky
}

      dice ?6 6      A turns 2 6' into random numbers between 1 and 6
2 5 Seven          A result was a 2 and 5 which sums to lucky 7
      dice ?6 6      A try again 2 random numbers between 1 and 6
2 1 Unlucky        A result this time was 2 and 1 which matches none of if's
      dice'' ?5p<6 6 A 5 sets(5p) of 2 6's(<6 6), random & check each('') set
2 3 Unlucky 2 2 Pair 6 4 Unlucky 2 3 Unlucky 3 4 Seven A 5 results

```

## PROBABILITY OF TWO DICE BEING EQUAL

Lets do 5 throws of 2 dice. To do this enclose(<)5 copies(p) of two 6's and let the ? turn all 5 pairs of 6's into random pairs of numbers 1-6:

```

      ?5p<6 6      A this is APL command and result is on next line
6 2 2 1 6 6 6 6 1 4 A we got five pairs of numbers(notice extra space
between each pair. Also notice we got two pairs (of 6's). To make APL
count matches we put an equal sign(=) between each pair(/'') like this.
      =/''?5p<6 6

```

```

0 0 1 1 0      A The ones tell us which pairs matched: (pairs 3 and 4)
Now lets add these 1's(with +/) getting 2 & divide by 5 to get the odds of
.4 Finally multiply by 100 to get 40 (for 40% matching pairs)

```

```
100×(+/''?5p<6 6)÷5
```

```
40      A so this time we got 40% matches (2÷5)
```

Now lets write a program to do this and call it DiceEqual.

```

DiceEqual←{100×(+/''?ωp<6 6)÷ω} A variable omega (ω) replaces 5
Now with ω we can try bigger samples and see if the real underlying
probability is indeed 40%. Lets just go for it with a million throws to get
a real good idea what the real probability is.

```

```
DiceEqual 1000000
```

```
16.6442      A looks like about 16.6% of time dice will match (not 40%).
```

Now lets try it 5 times with 100 throws each time(''):

```
DiceEqual''5p100
```

```
27 26 15 16 21      A got some variability between 15% and 27% matches
```

Now lets try it 5 times with 1,000,000 throws each time('')

```
DiceEqual''5p1000000
```

16.6033 16.6032 16.6488 16.6859 16.6377 A always got 16.60% to 16.69  
 From this we can see the advantage of large random samples. Large samples are less variable and they are more accurate. There are actually formulas that allow us to see the actual odds. The probability of two independent random events occurring together is simply the product of the probabilities of each event. In this case each die has 6 sides so the probability of getting say a 3 on one throw is 1/6 and the probability of any particular pattern such as "3 3" is  $1/6 \times 1/6 = 1/36$  which is 1 chance in 36. In our case we have 6 different ways to get a pair 1 1, 2 2, 3 3, 4 4, 5 5 and 6 6. So the odds of getting a matching pair is 6/36 which equals .1666666666. Looking back at our 5 1 million throws we can see that a sample size of 1,000,000 produces some pretty accurate results while the 5 size 100 samples were not so good. Just for fun lets try 1,000,000 throws 20 times and average them.

```
mean←{+/ω÷ρω} A mean program add up #'s(+/ω) and divide by n(ρω)
mean" (1 2 3)(8 6)(?1000ρ50) A mean each(")note:last=1000 rand# 1-50
2 7 25.015 A means for each group of numbers.
mean DiceEqual"20ρ1000000 A 20 groups of 1,000,000 pair throws
0.16662385 A took 17 seconds for my computer but is even more accurate.
```

Now lets see if the larger samples are less variable as suggested above by looking at some frequency plots. First I need a rounding function to round the percents to whole numbers so they can be put in categories. APL has the floor function([) which is useful here. But we can't just use the floor function because it always rounds down.

```
[1.2 3.4 1.8
1 3 1 A all numbers are rounded down, but we need 1.8 to be rounded up.
A solution is to add .5 to each number then use the floor([) function
[.5+1.2 3.4 1.8 A so the #'s become 1.7 3.9 2.3 and
1 3 2 A proper rounding is done. [1.7 3.9 2.3 is 1 3 2
So here is my round function. It is a little more general than needed here
so it can round to any number of decimal places by multiplying the number
by some magnitude of 10, adding .5, finding the floor then dividing it back
down by the same order of 10. It also has a default(α←0) which says to
round to 0 decimal places if nothing else is specified to the left.
```

```
round←{α←0 ⋄ ([0.5+ω×10*α)÷10*α} A define the round function
round 2345.45678 A default round to 0 (whole number)
2345
1 round 2345.45678 A round to 1 decimal place
2345.5
2 round 2345.45678 A round to 2 decimal places
```

2345.46

⌈2 round 2345.45678

A round to 100's place

2300

Next we need a program to put rounded results into categories:

```
freq←{↑(⌈u)(+⌈ω.=u←u[⌈u←uω])}
```

 A Here is the freq program:

freq finds unique values (uω) sorts them(⌈u), adds up all values(+⌈ω.=u) and combines them into a table.

Now we can do some plotting using the built in barchart icon. Lets create 500 10's(500p10) and send each(⌈) to DiceEqual which creates 500 random samples of size 10 of 2 dice tosses and calculates percentage of equal pairs for each of the 500 samples of size 10. The percentages are passed to round which rounds them to whole numbers and passes them to freq which counts up how many times each unique (u) percentage occurs and creates a table of the values and their frequencies passes this table to data where the values and their frequencies are stored. The plus sign(+) at the beginning of line displays 2 row data table that's stored in data

```
+data←freq round DiceEqual⌈500p10
```

 A call with 500 samples size=10

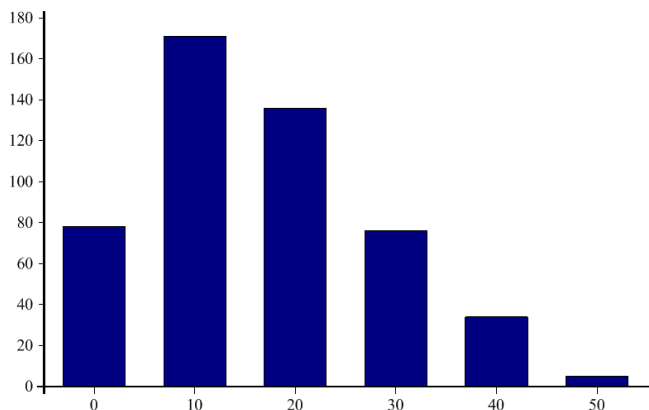
```
0 10 20 30 40 50
```

 A this row shows the percentages that occurred  

```
78 171 136 76 34 5
```

 A this row is frequency of percentage above it  
Now put cursor on data above & click Barchart icon at top of screen:

data



We have a range of 0% to 50% matching pairs, showing tremendous variability So 0% matches occurred 78 times 10% matches occurred 171 times etc.

Now lets try 500 samples of size 100

```
+data←freq round DiceEqual⌈500p100
```

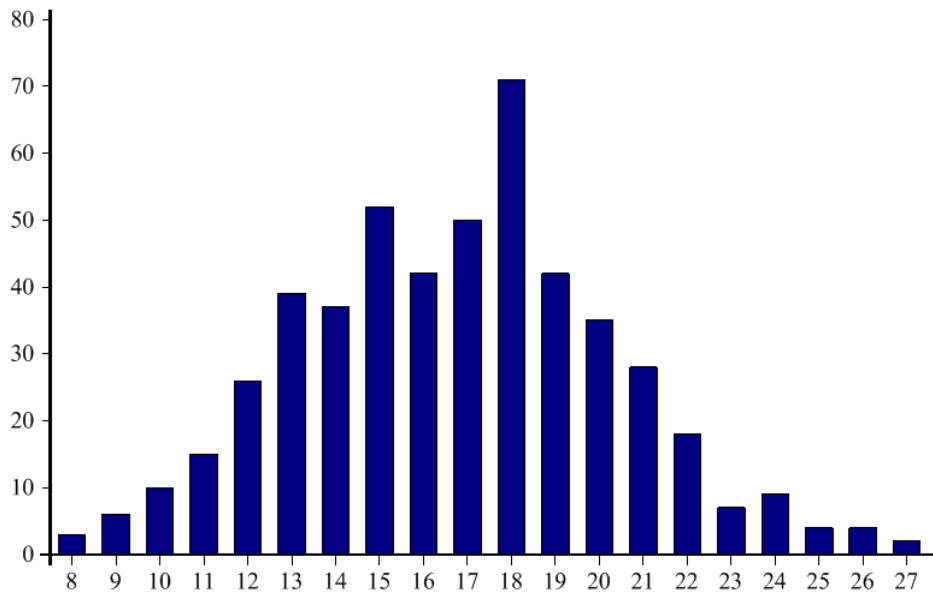
 A 500 samples of size 100

```
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
3 6 10 15 26 39 37 52 42 50 71 42 35 28 18 7 9 4 4 2
```

A Now put cursor on data above and click APL Barchart icon



data



A smaller range of 8% to 27% matching pairs but still lots of variability

Lets try 500 samples of 1,000

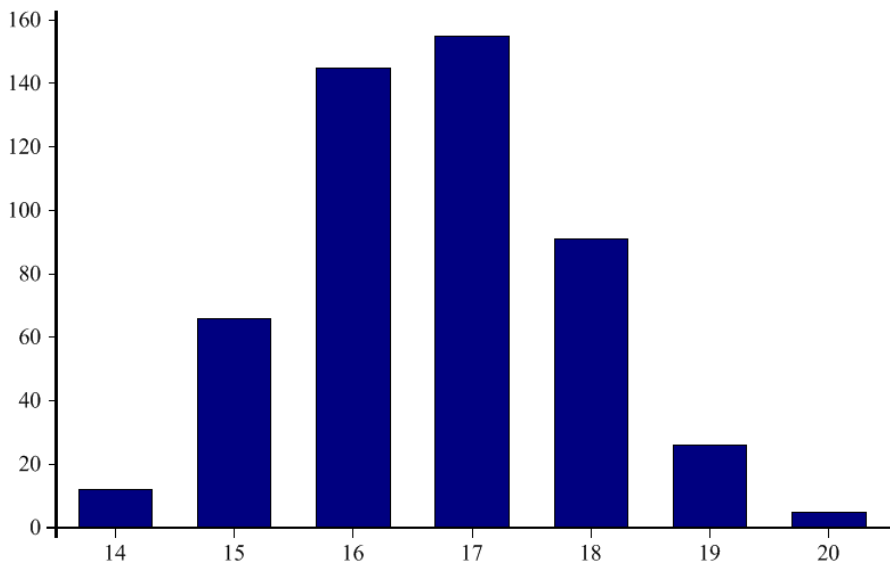
```
+data←freq round DiceEqual"500p1000
```

```
14 15 16 17 18 19 20
```

```
12 66 145 155 91 26 5
```

A Now put cursor on data above and click APL Barchart icon

data



Even smaller range of only 14% to 20% matching pairs. We are getting close

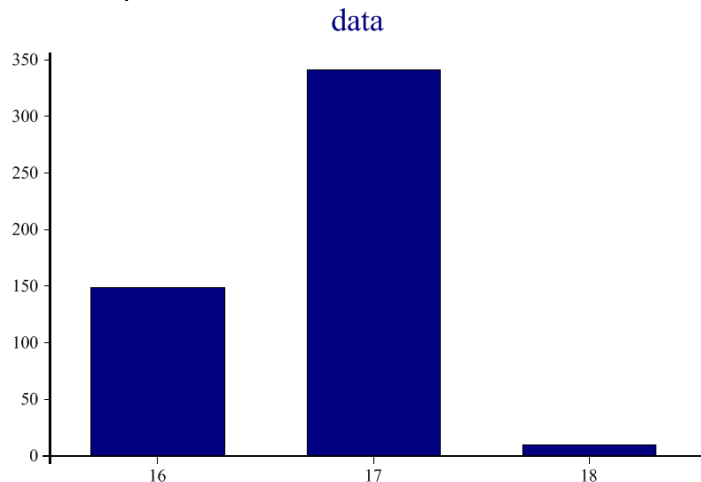
Lets try 500 samples of 10,000:

```
+data←freq round DiceEqual"500p10000 A 500 samples of size 10,000
```

```
16 17 18
```

149 341 10

A Now put cursor on data above and click APL Barchart icon



We have a range of only 16% to 18% matching pairs with almost none at 18 and more at 17 than 16. Thus we are zeroing in on the theoretical value of 16.66666. A sample size of 10,000 thus almost guarantees a close estimate of the true value. Good scientific research thus tries to get large sample sizes of possible for this reason. Sampling errors becomes a much smaller concern.

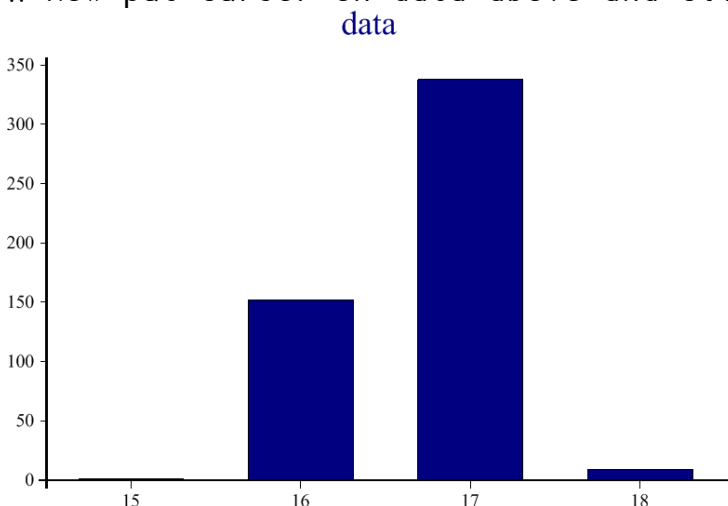
Lets try sample size 10,000 again to see if we'll have consistent results:

```
+data←freq round DiceEqual"500p10000 A 500 samples of 10,000 again
```

```
15  16  17  18
```

```
1  152  338  9
```

A Now put cursor on data above and click APL Barchart icon



We have range of 15% to 18% but there's only 1 in 15% and other frequencies are very very close. Replication is another important part of the scientific method in verifying that we are on the right track. Other things we could do to verify this result would be for you to try this on your

computer which may have a different random number generator or you could do the 500×10000 dice rolls yourself to check these results. ;)

## THE POWER OF 11

11 is an important number. It is used as a verification check for many things such as 10 digit book bar codes, overcoming skips or scratches on CDs and in all sorts of internet communications where static etc causes losses. By using 11 lost information can be figured out without having to retransmit the data.

Take a look at <http://www.numberphile.com/> and click on the 11-11-11 Eleven link.

In the book industry when 10 digit bar codes are used the 10 digits are always selected in a way so the check number is evenly divisible by 11. This is explained on the video link above. Here is an example:

Here is a barcode: 0 3 1 2 1 5 2 2 7 2 from book Tongue-Fu by Sam Horn.

Each of these numbers is multiplied by a number from 10 to 1.

0	3	1	2	1	5	2	2	7	2	bar code
10	9	8	7	6	5	4	3	2	1	numbers from 10 to 1
-----										
0	27	8	14	6	25	8	6	14	2	resulting multiplication

The sum of (0 27 8 14 6 25 8 6 14 2) is 110 which is divisible by 11: (110÷11=10) . All 10 digit barcodes on backs of books when multiplied like this and added up are divisible by 11. This is called the checksum.

Here's how to do this in APL. First enter the program like this:

```
barcode11←{0=11|+/\ω×⊢10}
```

and test it like this:

barcode11 0 3 1 2 1 5 2 2 7 2	A good bar code
1	A 1 means good, 0 would be bad

The computer returns a 1 for yes if it is divisible by 11. A bad barcode will result in a 0.

```

barcode11 7 3 1 2 1 5 2 2 7 2      A bad barcode
0                                     A 0 means bad, 1 would be good

```

Here is how it works from right to left: The program  $\{0=11|+/\omega\times\phi\iota10\}$  generates the numbers 1-10( $\iota10$ ), reverses them ( $\phi$ ) and multiplies the reversed numbers( $10-1$ ) by  $\omega$ (which is the barcode read into the program) then sums the resulting numbers up( $+/$ ) and finds the residue or remainder( $|$ ) of division by 11. If the residue equals( $=$ ) 0 that means the sum is evenly divisible by zero with nothing left over(no residue) and the program returns a 1(if true that  $0=\text{the residue}$ ) or 0(if  $0\neq \text{the residue}$ )

Here is an example using residue( $|$ ):

```

13|26 28 30      A remainder(|) of 13 divided into each # 26 28 30
0 2 4      A 13 into 26 has no remainder. 13 into 28 residue is 2 and 30 is 4

```

Now I was curious how good this barcode check was so I tested it by taking a valid(divisible by 11) bar code and randomly changing 1 number and checking the new number to see if would indeed fail the divide by 11 check.

I wanted to check it in a lot of ways to be certain this barcode method would catch all slight changes, so I wrote a program to randomly change one number in a 10 digit bar code. Here is my program:

```

change1←{c[i]←(( $\neg$ 1+ $\iota$ 10)~((i←?10)▷c←ω))[?9] ♦ c}

```

Here's how it works. First there are two commands separated by diamond( $\diamond$ ).

$c[i]←((\neg 1+\iota 10)\sim((i\leftarrow ?10)\triangleright c\leftarrow \omega))\leftarrow [?9]$  This part determines a random number to insert into the  $i^{\text{th}}$  position( $c[i]\leftarrow$ ) of my changed string  $c$ . First the changed string is created by copying the old string ( $c\leftarrow \omega$ ). Next, a random position to change( $i$ ) between 1-10 is made by  $(i\leftarrow ?10)$ . The code: $(\neg 1+\iota 10)$  gets the numbers 1-10 and adds a negative 1( $\neg 1$ ) to each resulting in the numbers 0-9. The  $\sim((i\leftarrow ?10)\triangleright c)$  part finds the value currently in position  $i$  of  $c$  and eliminates( $\sim$ ) it from the numbers 0-9 found by: $(\neg 1+\iota 10)$  so I am left with 9 possible new numbers to insert in  $c[i]$ . The  $[?9]$  part selects one of these 9 new numbers which is placed in  $(c[i]\leftarrow)$ .

$c$  by itself after the diamond( $\diamond$ ) simply tells the program to return the entire changed barcode( $c$ ) back to be displayed when the program is called:

```

x←0 3 1 2 1 5 2 2 7 2 A for convenience store good barcode in x
change1 x
0 3 1 2 6 5 2 2 7 2      A #1 random change 5 digit to 6
change1 x
0 3 1 2 2 5 2 2 7 2      A #2 random change 5 digit to 2(same pos)
change1 x
0 3 1 2 1 5 2 2 2 2      A #3 random change 9 digit to 2
change1 x
0 3 1 2 6 5 2 2 7 2      A #4 random change 5 digit to 6(same as #1)

```

Now I can check these to see if they fail the divide by 11 check.

```

barcode11 0 3 1 2 6 5 2 2 7 2
0

```

The zero means it failed the check. Indeed all these 1 digit changes fail the check. This is promising but I need to do much more checking to be sure so I need to simplify things some more to get more efficient.

First I can put the two programs together to check more quickly like this:

```

barcode11 change1 x
0
change1 changes 1 number of barcode in x and then barcode11 checks that #

```

If I wanted to see the change and check it too I could insert a `□`.

```

barcode11 □←change1 x
0 3 1 2 7 5 2 2 7 2      A this is x with one number changed(7)
0                          A it fails the check.
This shows the changed code and that it failed the check.

```

However, this is still not a very extensive check, so I did the following which does 100,000 changes on the string(x) and adds up how many pass the check. The result was zero, meaning none of the changes pass the check, so I feel pretty confident that the 11 barcode check method is a good one. Here is the program that does the 100,000 check.

```

+ /barcode11"change1" 100000p<x
0                          A none of the 100,000 new strings passes check

```

Here is how this works. First I made up 100,000 x strings with the same valid barcode. The enclose (c) symbol takes the 10 digit string(x) and puts in a packet size of 1 and then 100000p makes 100000 of these packets. The each operator (") tells the programs to operate on each of the 100,000 x string packets. The change1 program grabs each(") of these same good string packets and makes one random change in each and passes it to the barcode11 program which checks each(") of the 100,000 new string packets and returns a string of 100,000 0's and 1's indicating if each changed string passed the divide by 11 check. Finally the string of 100,000 0's and 1's is added up (+/) and the result is zero which is displayed and tells us none of the 100,000 random changes was valid.

## MORTGAGE CALCULATIONS

See sample problem from Wikipedia

[http://en.wikipedia.org/wiki/Amortization\\_schedule](http://en.wikipedia.org/wiki/Amortization_schedule)

```
Setup:P=loan:$100000 i=yearly interest rate:7% n=#payments:20yrs*12months
      P←100000 ♦ i←.07÷12 ♦ n←20*12 A assign values
      PaymentMonthlyAnuityFormula←{P i n←ω ♦ P×i+i÷((1+i)*n)-1} A define
      PresValOfAnuity←{A i n←ω ♦ (A÷i)×1-1÷(1+i)*n} A define
```

Test:

```
+MP←PaymentMonthlyAnuityFormula P i n A call program
775.2989356 A result is MP the total monthly payment
```

```
PresValOfAnuity MP i (12*20-7) A call program for 7 years in
79268.01945 A principal amount owed after 7 years of payments
```

```
P×i A calculate Init payment to interest (Principle × interest rate)
583.3333333
```

```
MP-583.33 A Calc Init pay to Prin (monthly paym - paid to interest)
191.97
```

```
(P-191.97)×i9 A 2nd pay to int (principle - prev principle payment)
582.21
```

```
MP-582.21 A 2nd payment to Principle
193.09
```

```

Program:monthly payments table=Interest,Principle & Remaining Principle
tbl←amort(P i n);MP;int;k;pri
    A amortization schedule
MP←PaymentMonthlyAnuityFormula P i n          A monthly payment
tbl←1 4p'Period' 'Interest' 'Principle' 'Balance' A column titles
:For k :In n          A loop n times k changes from 1 to n
    P←pri+MP-int←P×i    A calc int principle(pri) and new balance(P)
    tbl,←k,2⌘int,pri,P A add row of data(k,int,pri,P) to table
:EndFor    A end of loop got back to :For

```

To create this fns type: )ed amort press enter and type lines into editor.

Call the program to create amortization table for above values of: P,i,n

amort P i n A program creates amortization table with 240 rows

Period	Interest	Principle	Balance
1	583.33	191.97	99808.03
2	582.21	193.09	99614.95

..... A rows deleted for brevity

239	8.97	766.33	770.80
240	4.50	770.80	0.00

## ROOTS OF A POLYNOMIAL

Given an equation such as  $y=2x^2 +1x -10$ . what are it's roots(the x values that cause y to be equal to 0). Here is an APL program to find them:

```
quad←{a b c←ω ⋄ d←(b*2)-4×a×c ⋄ (+/x),-/x←((-b),d*.5)÷2×a}
```

```

quad 2 1 -10          A try program with equation: y=2x² +x -10
2 -2.5                A so if x=2 or -2.5 the equation for y = 0

```

to check the result substitute 2 and -2.5 into the equation

```

x←2 -2.5              A store roots in x
(2×x*2)+x+10          A test the equation with values of x.
0 0                   A 0 0 result so 2 & -2.5 are roots of eq.

```

The above is clear but there is an even easier way in APL.

APL has a special symbol to insert values into equation of this general type. It will also work for higher order equations like  $3x^5+2x^3+x^2+5$ . For this equation if  $x←2 -2.5$  then  $(x⌈⋄c3 0 2 1 5)$  would do it. This makes it

very easy to make y values from the x values or to test to be sure the roots found are correct.

```

2 -2.5 1 -10      A or x1 -10
0 0               A 0 0 result so 2 & -2.5 are roots of eq.

```

But not all equations have 2 roots, some equations have only one root and others have only imaginary roots. Here are two APL program to calculate any of these possible cases the first labels the result the second just returns the roots which can then be passed on to other APL programs. How many roots there are can be determined by the sign(x) of the calculation of disc. If sign of disc=1(positive) there are two real roots, if sign of disc=0(zero) there is one real root and if sign of disc=-1 there are two imaginary roots. Here is the complete program with labeled output for the 3 cases:

```

QUAD (a b c);disc;u;v
:Select xdisc←(b×b)-4×a×c      A xdisc is sign(1 0 or -1) of disc
:Case 1 ♦ u←-b÷2×a ♦ v←(disc×0.5)÷2×a
      'Two Real Roots:',(u+v),'and',(u-v)
:Case 0 ♦ 'One Real Root',-b÷2×a
:Case -1 ♦ u←-b÷2×a ♦ v←((-disc)×0.5)÷2×a
      'Two Complex Roots',(u,'+',v,'I'),' and',(u,'-',v,'I')
:EndSelect

```

To create this fns type )ed QUAD press enter & type lines into editor. Lets test it out with the same example then with 2 other equations:

```

QUAD 2 1 -10      A      A 2x2 +x -10
Two Real Roots: 2 and -2.5
QUAD 3 -2 10      A 3x2 -2x +10
Two Complex Roots 0.333333 + 1.795054 I and 0.333333 - 1.795054 I
QUAD 9 12 4       A 9x2 +12x +4
One Real Root -0.6666666667

```

Here is a more complete version of page 11 quad that returns all roots but also unlabeled. This will be more useful to pass to other programs:

```

quad←{
[1]      a b c←w ♦ d←(b*2)-4×a×c      A input w to a b c ♦ find disc d
[2]      d>0:ϕ(+/x),-/x←((-b),d×0.5)÷2×a A if disc>0 show 2 roots
[3]      d=0:-b÷2×a                    A if disc=0 show 1 root
[4]      d<0:θ                          A if disc<0 show nothing(θ)
}

```



Lets try it with the same 3 equations. I have used a built in APL utility ]disp to display the results so you can see their structure. ]disp is used for display only, not when passing results to other programs.

```
]disp quad "(2 1 -10)(3 -2 10)(9 12 4)"
```

```

┌──────────┬──┬──┬──┐  A
├─2.5 2┤0├─0.6666666667┤  A 2 roots, no roots, 1 root
└──────────┴──┴──┴──┘

```

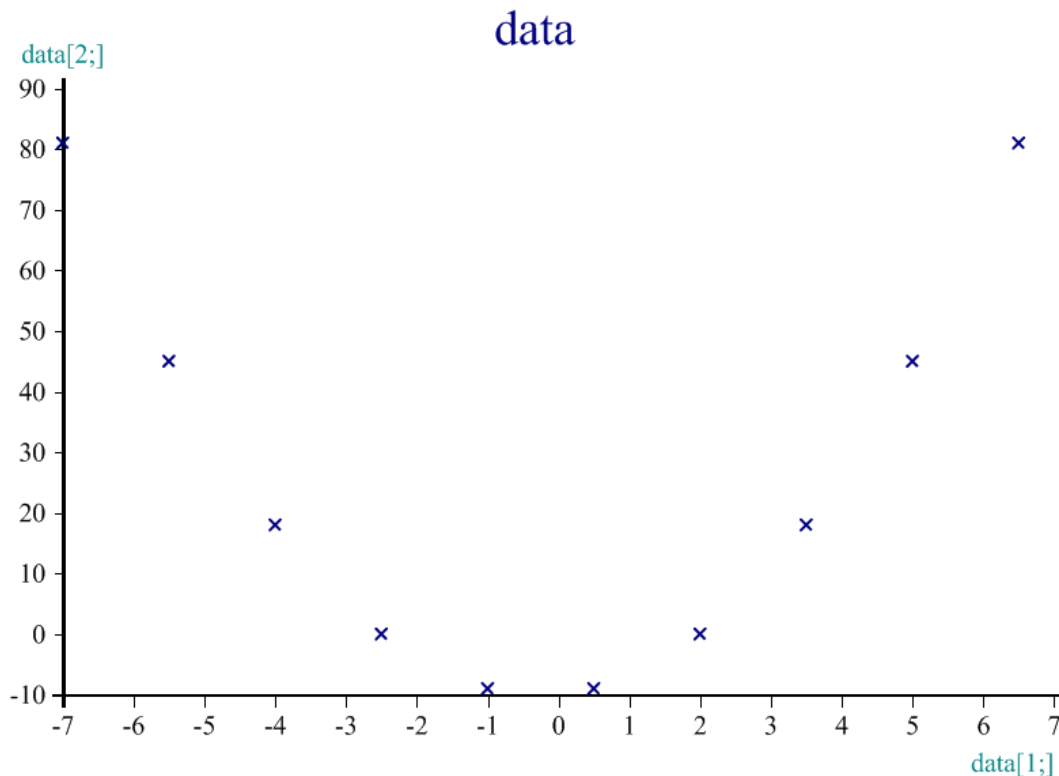
Now lets plot equation  $2x^2 + x - 10$  so we can see its shape and where the roots are. First we need to generate some x plotting values around the roots of -2.5 and 2 so we can see these critical points clearly in the upcoming plot. The program xaroundroots below does that. It takes the two roots as input on the right and the number of x values to make on the left. It then finds the difference(dif) between the two roots and generates  $\alpha(10)$  x values from the lower root(d) minus the difference to the upper root(u) plus the difference so in this case the difference between roots -2.5 and 2 is 4.5 so 4.5 is subtracted from -2.5 giving -7 which is the first x value as can be seen below. Then it takes the upper root which is 2 and adds 4.5 to that giving 6.5 which is the highest of the 10( $\alpha$ ) x values.

```

xaroundroots←{dif←(u←[ /ω)-d←[ /ω A find difference between 2 roots
du←(d,u)+(-dif),dif A - & + dif roots finds low & high values(du)
(1>du)+(-1+ια)×(-/ϕdu)÷α-1 A make α x values in range(1>du to 2>du)
} A to create this fns type: )ed xaroundroots press enter and type lines
2⍉x←10 xaroundroots ⍎←quad 2 1 -10 A show roots & make 10 x values
2 -2.5
-7.00 -5.50 -4.00 -2.50 -1.00 0.50 2.00 3.50 5.00 6.50
2⍉y←(2×x*2)+x+-10 A put above x values into equation to get y's
81.00 45.00 18.00 0.00 -9.00 -9.00 0.00 18.00 45.00 81.00
2⍉data←y,[.5]x A put the x and y values into a matrix for plotting.
81.00 45.00 18.00 0.00 -9.00 -9.00 0.00 18.00 45.00 81.00
-7.00 -5.50 -4.00 -2.50 -1.00 0.50 2.00 3.50 5.00 6.50

```

Now just click on word data above then click on scatterplot icon to plot:



So now lets put this together in a little program so we can do it easily:  
`rootsandplot←{α←100 ♦ x←α xaroundroots □←quad ω ♦ y←x12+x-10 ♦ plotxy x y}`

Notes: this program really has 4 lines separated by diamonds (♦)

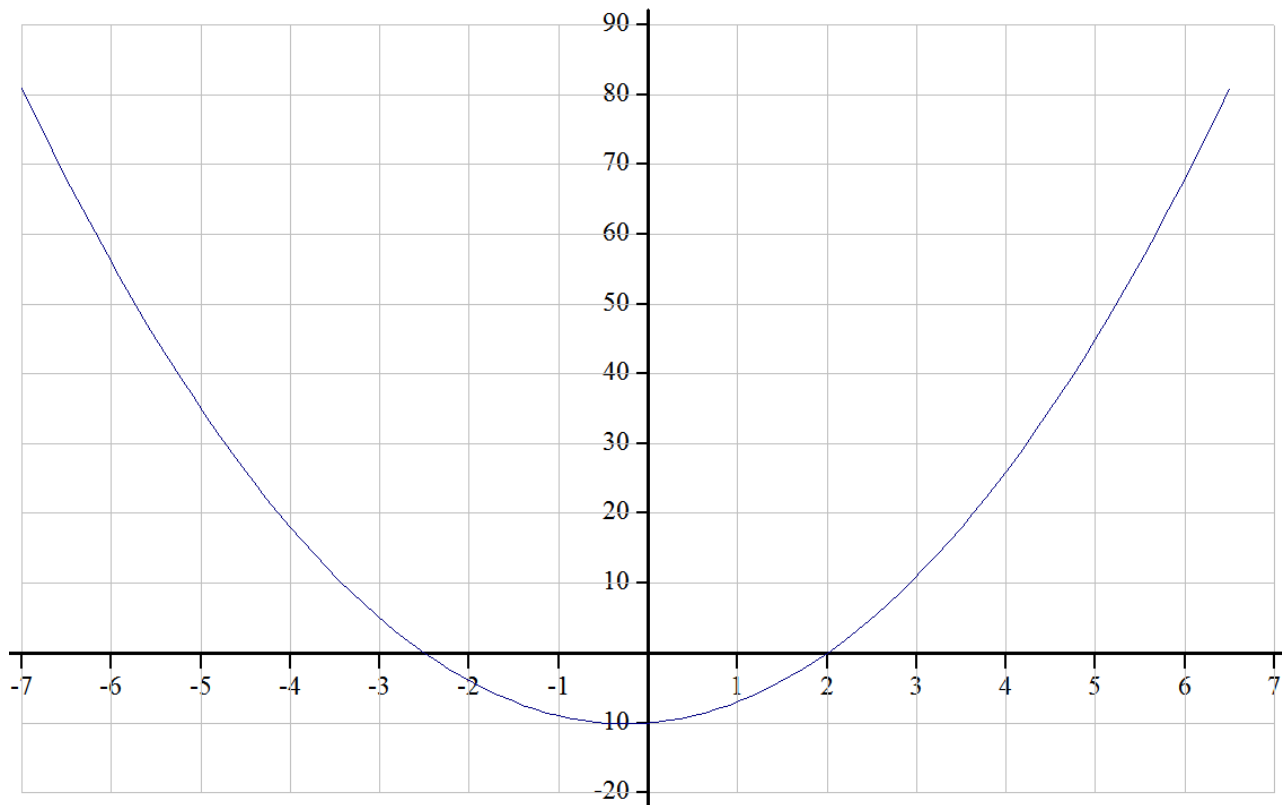
1.  $\alpha \leftarrow 100$  sets default to make 100 x & y values. If you don't specify a number on the left when you call the program you will get 100 x & y values.
2.  $\square \leftarrow$  displays roots computed by quad program using input equation ( $\omega$ ) and then passes the roots to xaroundroots which finds 100 x values near the roots.
3.  $y \leftarrow x^2 + x - 10$  puts the found x values into the equation (i.e.  $\omega = 2x^2 + x - 10$ ). As mentioned above this tricky code is APL equivalent to  $y \leftarrow (2 \times x^2) + x - 10$
4. finally plotxy passes x and y to little plot program I wrote to make a prettier display than above. Here it is:

```
R←{ax0}plotxy data
[1]  A plot data:x=col1 y=col2 or x=vector1 y=vector2
[2]  ax0←0=□NC'ax0' A if no ax0 axes cross at 0
[3]  :If 2≡data ♦ data←Q↑data ♦ :End
[4]  ch.Set'Lines' 1 2 4 5
[5]  ch.Set''(ax0,ax0,1)/('Xint' 0)('Yint' 0)('XYPLOT,GRID')
[6]  ch.Plot data ♦ PG←ch.Close
[7]  R←'View PG A to see it'
```

To create this fns type )ed plotxy press enter and type lines into editor.

Now lets the try program rootsandplot for the equation:  $2x^2 + x - 10$

```
rootsandplot 2 1 -10  A call program shows roots and plots xy data
2 -2.5                A shows roots. Plots 100 xy value pairs
View PG A to see it   A just press enter on this line to see plot
```

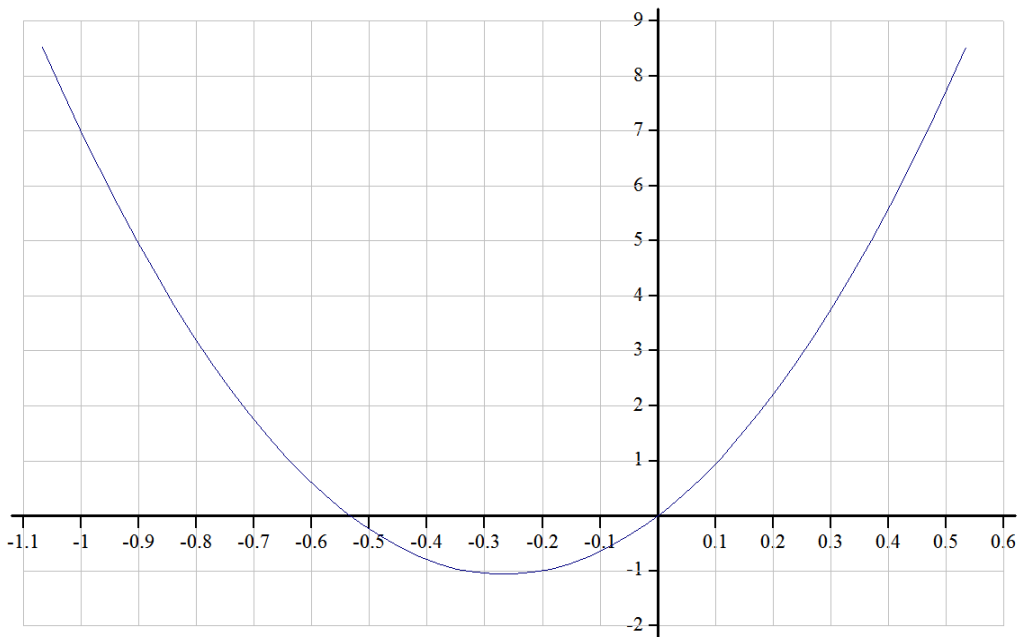


Notice where the roots( $y=0$ ) 2 and  $-2.5$  are on the plot.

This program will plot data for any polynomial with 2 real roots simply by entering the parameters for  $x^2$   $x^1$  and  $x^0$ .

Now lets try:  $y=15x^2 + 8x + 0$  and request only 50 values to plot.

```
50 rootsandplot 15 8 0  A plot 50 xy points for y=15x^2 + 8x + 0
0 -0.5333333333333333  A shows 2 roots.
View PG A to see it     A just press enter on this line to see plot
```



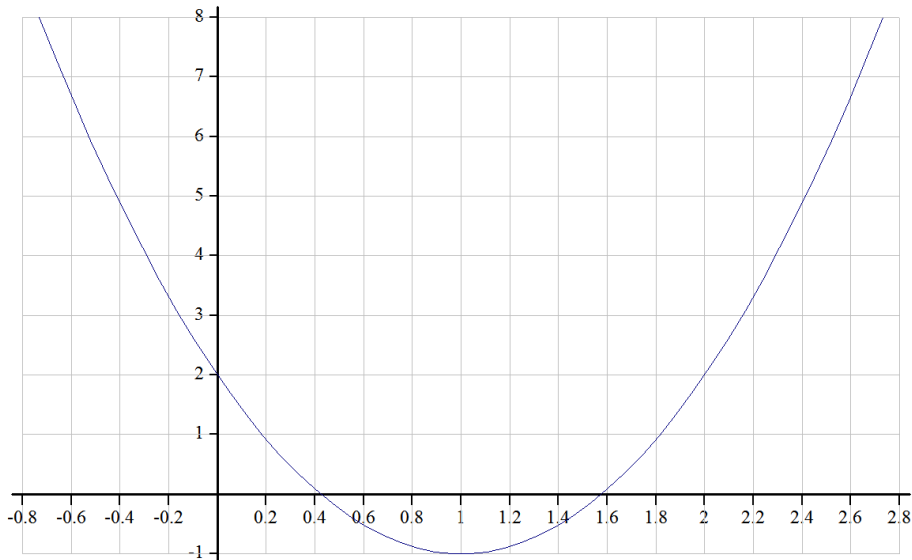
Again notice where roots are and see that xaroundroots centers plot nicely

## QUADRIC EQUATIONS AND FUNCTIONS

Quadratic equations are of the general form  $y = ax^2 + bx + c$ . The above program only works when there are two roots and it does not tell us either vertices or minimums of the function. So here is a more complete program. Lets plot such an equation in APL. Lets try  $y = 3x^2 + \sqrt{6}x + 2$ . Here is a way to plot such an equation in APL by entering just a b and c into the program.

QuadPlot 3  $\sqrt{6}$  2

View PG 9 to see it



$y = ax^2 + bx + c$  a= 3 b=  $\sqrt{6}$  c= 2 Function min at x y= 1.000  $-\sqrt{1.000}$  xintercepts= 0.423 1.577

The program footnote tells us that the function has a minimum at  $x=1$   $y=-1$  and the it crosses the x axis twice, once at .423 and again at 1.577.

Here is the QuadPlot program:

```

▽ QuadPlot(a b c);x;y;xint;xvert;yvert;mm;rng;ft
  A Plot quadratic eq: QuadPlot 2 -1 -7 for: 2x*2 -x -7 (a=2 b=-1 c=-7)
  mm←((1+a<0)='F min' 'F max'), ' at x y=' A "max" if a<0 otherwise "min"
  xvert←-b÷2×a A xvert=where y is min or max
  yvert←xvert1''c a b c A solve eq for yvert using xvert
  xint←,quad a b c A quad formula for x intercepts
  rng←±(1+pxint)='0,xvert' '0,xint' 'xint' A find range of x values to plot
  :If rng≡0 0 ◇ rng←-10 10 ◇ :End A fix range if at 0 0
  x←xaroundroots rng A find good x values to plot
  y←x1''c a b c A solve eq for y using x values
  ft←'y=ax*2+bx+c a=' 'b=' 'c='mm'xintcepts=',',a b c(3×xvert yvert),c3×xint
  ch.Set'Footer' ft
  plotxy x y

```

To create this fns type )ed QuadPlot press enter & type lines into editor.

## BASIC STATISTICS:

mean←{(+/ω)÷(pω)}	A sum(+/) of #'s divided by #(p) of #'s
max←{[/ω]}	A maximum of numbers([/])
min←{[/ω]}	A minimum of numbers([/])
range←{(max ω)-(min ω)}	A range - max minus min of numbers
sort←{ω[Δω]}	A sort numbers up(Δ). Ψ would sort down
median←{mid←(1+ps←sort ω)÷2 ◇ mean s[(mid)([mid])]}	

The median is defined as the middle number if there are an odd # of #'s and the average of the two middle numbers if there are an even # of numbers. The above median fns first sorts the data into s and finds the midpoint which is either a whole number position for odd # of #'s or a position 1/2 way between the two middle positions(mid) if there are an even number of numbers. After the diamond(◇) the fns averages the 1 or two middle numbers s[(mid)([mid)]. Lets try a couple to see how it works.

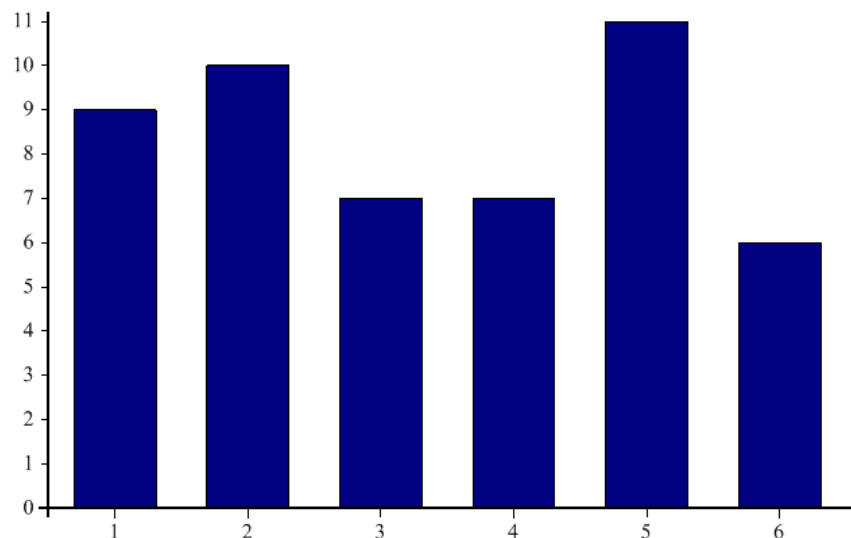
median 5 6 8 7	A s=5 6 7 8, mid=2.5, [mid]=2, [mid]=3
6.5	A s[2 3]=6 7, average of 6 7=6.5
median 9 5 8	A s=5 8 9, mid=2, [mid]=2, [mid]=2
8	A s[2 2]=8 8, average of 8 8=8

```

freq←{↑(⌊"u)(+⌊ω°. =u←sort uω)} A frequency count fns
+freqs← freq num←?50p6 A call freq with 50 rand(?) #'s(1-6) in num
1 2 3 4 5 6 A here are the label #'s in row 1 of freqs
9 10 7 7 11 6 A here are the frequencies in row 2 of freqs
The freq function: u is sorted unique(u) values of the 50 rand #'s(num)
#'s are counted into unique categories 1-6 (+⌊ω°. =u) & labeled(⌊"u)
Now click first on freqs then on Barchart icon at top to see:

```

freqs



```

mode←{↑(c(f>1ϕf)^(f>~1ϕf))/~v f←0,"(↓freq ω),"0} A mode uses freq
mode num A call mode program with same num used above for freq.
2 5 A two modes one at 2 one at 5
10 11 A mode at 2 has a frequency of 10. mode at 5 has frequency of 11
modes are defined as any frequencies that are higher than frequencies
immediately before or after it or are at either end and are higher than the
one frequency that is either before it or after it. The program finds the
frequencies in this case for values 1-6:9 10 7 7 11 6 and puts 0's before
and after the frequencies then compares by rotating(ϕ) f to values before
and(^) after and if it greater than both it is a mode as can be seen here:
↑v f (1ϕf) (~1ϕf) A This displays v f 1ϕf & ~1ϕf as a 4 row table
0 1 2 3 4 5 6 0 A v are the values with 0's on each end
0 9 10 7 7 11 6 0 A f 10 & 11 only ones greater than(>) 1ϕf~1ϕf
9 10 7 7 11 6 0 0 A 1ϕf
0 0 9 10 7 7 11 6 A ~1ϕf

```

## LINEAR REGRESSION: COMPUTE LINE FROM RAW DATA

sd corr LinReg LinRegPlot

(see page 332 of Algebra 1 book) see also ch.Set 'Order'

Programs:sd:standard deviation corr:correlation Reglin:linear regression

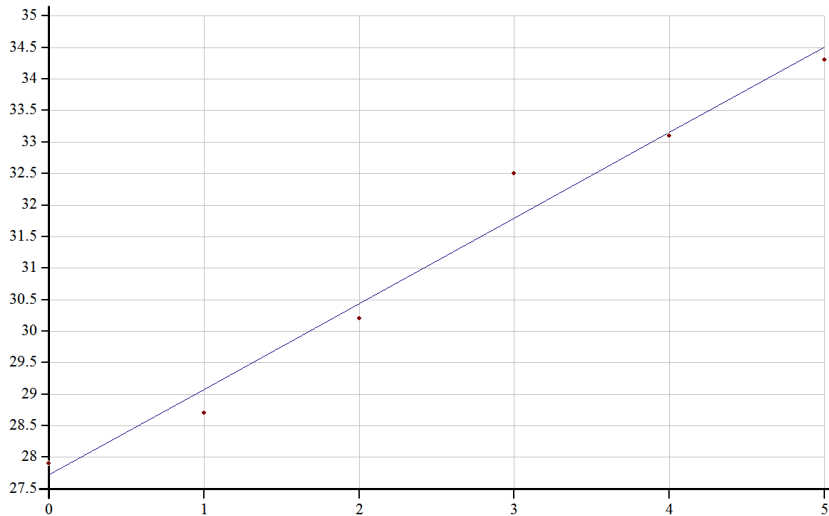
```
sd←{((+/(ω-mean ω)*2)÷~1+ρω)*0.5} A define program for standard deviation
corr←{ma mw←mean`α ω ◇ (+/(α-ma)×(ω-mw))÷((+/(α-ma)*2)*0.5)×((+/(ω-
mw)*2)*0.5))} A define program for correlation
RegLin←{'y=ax+b a=' 'b=' 'r=' 'r*2=',''(ω⊖α°.*1 0),(α corr ω)*1 2}
```

```
R←x RegLinPlot y;yline;foot;a;b A define linear regression plot
[1] ch.Set'Head' 'Linear Regression Plot'
[2] a b←y⊖φ↑1,`x A determine regression line formula
[3] yline←(a×x)+b A get regression line points
[4] ch.Set'Footer'(x RegLin y) A get eq,r r*2 for footer label
[5] ch.Set'XYPLOT,GRID' A set up the plot
[6] ch.Plot⊖↑x yline A plot regression line
[7] #.ch.SetMarkers'Bullet' A ch.Δmarkers shows other symbols
[8] ch.Scatter⊖↑x y A data points as Bullets
[9] PG←ch.Close
[10] R←'View PG A to see it'
```

To create this fns type )ed RegLinPlot press enter & type lines into editor.

```
x←0 1 2 3 4 5 A x raw data
y←27.9 28.7 30.2 32.5 33.1 34.3 A y raw data
x RegLinPlot y A do regression
View PG A to see it
```

## Linear Regression Plot



y=ax+b a= 1.357142857 b= 27.72380952 r= 0.9881725632 r\*2= 0.9764850146

$$y = 1.36x + 27.7 \text{ (corr=.99)}$$

If you had only two points to plot this program would just find the perfect line equation between the two points and the correlation would be 1.0. The Domino (Ⓜ) used in line [2] above is very powerful. It can be used to solved multiple regression problems where you are fitting multiple sets of data and nonlinear regression. It can also be used to solve sets of simultaneous equations.



### 5.2.3 - Solving a Set of Equations

Here is a set of three linear equations with three unknowns  $x$ ,  $y$ , and  $z$ , written using traditional mathematical notation:

$$-8 = 3x + 2y - z$$

$$19 = x - y + 3z$$

$$0 = 5x + 2y$$

This set of equations can be represented using a vector for the constants and a matrix for the coefficients of the three unknowns, as shown below:

$$\begin{bmatrix} 3 & 2 & -1 \\ 1 & -1 & 3 \\ 5 & 2 & 0 \end{bmatrix} \text{ Coefs} + \begin{bmatrix} -8 \\ 19 \\ 0 \end{bmatrix} \text{ Cons} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ XYZ}$$

To solve the above set of equations, we must find a vector of three values  $XYZ$  such that:

$$\text{Cons} \text{ is equal to } \text{Coefs} \times XYZ$$

We can find such a solution provided that the matrix **Coefs** has an inverse, i.e. that it is non-singular.

Let us multiply both sides of the equation by the inverse of **Coefs**:

$$\begin{aligned} \text{If } \text{Coefs} \times XYZ & \text{ is equal to } \text{Cons} \\ \text{then } (\text{InvCoefs}) \times \text{Coefs} \times XYZ & \text{ is equal to } (\text{InvCoefs}) \times \text{Cons} \end{aligned}$$

Knowing that  $(\text{InvCoefs}) \times \text{Coefs}$  gives the identity matrix (let's call it **I**), the expression can be reduced further:

$$\begin{aligned} \text{Since } (\text{InvCoefs}) \times \text{Coefs} \times XYZ & \text{ is equal to } (\text{InvCoefs}) \times \text{Cons} \\ \text{then } \text{I} \times XYZ & \text{ is equal to } (\text{InvCoefs}) \times \text{Cons} \\ \text{and consequently } XYZ & \text{ is equal to } (\text{InvCoefs}) \times \text{Cons} \end{aligned}$$

Eureka! We found a way of calculating the values we had to find:

$$\begin{bmatrix} 2 & -5 & 4 \end{bmatrix} \times XYZ = \begin{bmatrix} -8 \\ 19 \\ 0 \end{bmatrix} \times \text{InvCoefs} \Rightarrow \text{You can check. This is correct!}$$

More generally:

$$\text{Solutions} = (\text{Inv Coefficients}) \times \text{Constants}$$

Note that in the formula above we multiply *Constants* by the inverse (or reciprocal) of a matrix. Multiplying by the reciprocal of something is usually known as division, so perhaps this is true here as well? Yes it is, and we'll show that in the next section.


The dyadic form of *Domino* implements matrix division, so it can do exactly what we have just done: It can easily solve sets of linear equations like the one shown above:

$2^{-5} 4$  **Cons**  $\otimes$  **Coefs**  $\Rightarrow$  Equivalent to  $(\otimes \text{Coefs}) + \cdot \times \text{Cons}$   
 $\Rightarrow$  We found the same solution as before.

Naturally, this method works only if the coefficient matrix has an inverse. In other words, the set of equations must have a single solution. If there is no solution, a DOMAIN ERROR will be reported.

We can summarise this as follows:

Given a system of  $N$  linear equations with  $N$  unknowns, let the matrix of the coefficients of the unknowns be named *Coefficients*, and the vector of constants be named *Constants*, the system can be solved using matrix division like this:

*Solutions* ← *Constants*  *Coefficients*

# LINEAR QUADRATIC AND CUBIC REGRESSION

First some data for x and y

```
x ← -2 -1 0 1 2  ◇  y ← 0.25 0.5 1 2 4
```

```
RegLin←{('y←(a*x)+b'),('a←' 'b←' 'r=' 'r*2=',⌞⌞(ω⊖α◦.*1 0),
      (α corr ω)*1 2)}      # define linear regression function
```

x RegLin y      a call Linear Regression

```
y←(a×x)+b  a← 0.9  b← 1.55  r= 0.93  r*2= 0.87  a linear results
```

```
RegQuad←{(c'y←(a*x*2)+(b*x)+c'),('a←' 'b←' 'c←','⌘'(ω⊖α◦.*2 1 0))}
x RegQuad y           A call quadratic regression function
```

```
y←(a×x*2)+(b×x)+c    a← 0.29    b← 0.9    c← 0.98    A quadratic results
```

```
RegCube←{(c'y←(a×x*3)+(b×x*2)+(c×x)+d'),
          ('a←' 'b←' 'c←' 'd←',''⌈''(ω⊞α°.*3 2 1 0))}
x RegCube y
```

A call cubic regression function

```
y ← (a × x × 3) + (b × x × 2) + (c × x) + d    a ← 0.06    b ← 0.29    c ← 0.69    d ← 0.98
```

Notice the similarity in the above 3 functions.

Regression coefficients	Equation	APL Domino Operator
Linear: a b	$y \leftarrow (a \times x) + b$	$\omega \boxminus \alpha \circ . * 1 \ 0$
Quadratic: a b c	$y \leftarrow (a \times x^2) + (b \times x) + c$	$\omega \boxminus \alpha \circ . * 2 \ 1 \ 0$
Cubic a b c d	$y \leftarrow (a \times x^3) + (b \times x^2) + (c \times x) + d$	$\omega \boxminus \alpha \circ . * 3 \ 2 \ 1 \ 0$

But there is a simpler way in APL. Since the 3 programs are so similar it is possible to write one function that can do linear quadric and cubic and actually it can go beyond cubic if you wish. Here is the function:

```
reg←{x y←ω ⋄ y⌶x∘.*ϕ-1+ι1+α} ρ does all types of regressions
1 reg x y                ρ 1 is x1 linear regression
0.9 1.55
2 reg x y                ρ 2 is x2 quadratic regression
0.29 0.9 0.98
3 reg x y                ρ 3 is x3 cubic regression
0.06 0.29 0.69 0.98
(ι3) reg∘ c x y ρ do linear, quadratic cubic all at once
0.9 1.55 0.29 0.9 0.98 0.06 0.29 0.69 0.98
```

If you wanted little better labeling of these equations pass them to the RegEq function

```
RegEq←{1↓⊃,/((⊂'+''),∘((⊂∘ω),∘(⊂'×x*'))),∘(⊂∘ϕ-1+ιρω),∘'')'}
↑RegEq∘ (ι3) reg∘ c x y
(0.9×x*1)+(1.55×x*0)                ρ linear
(0.2857142857×x*2)+(0.9×x*1)+(0.9785714286×x*0) ρ quadric
(0.0625×x*3)+(0.2857142857×x*2)+(0.6875×x*1)+(0.9785714286×x*0) ρ cubic
These lines could be easily executed passed and compared in plotxy.
```

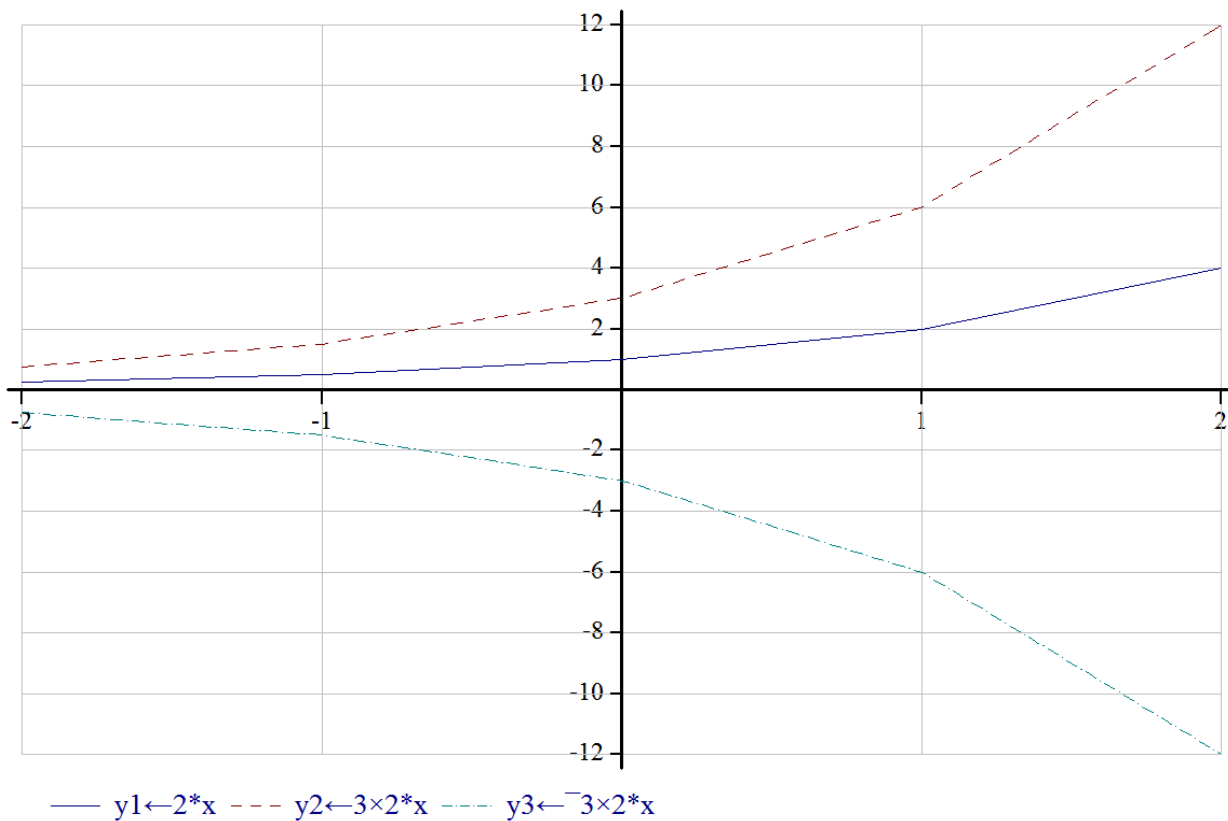
## PLOTTING 3 EXPONENTIAL FUNCTIONS TO COMPARE

Here is some data for three exponential functions from page 521 of McDougal Littell Algebra I which I will show you how to plot.

$x←^{-3+ι5}$	$y1←2*x$	$y2←3*2*x$	$y3←^{-3*2*x}$
<sup>-2</sup>	0.25	0.75	
<sup>-1</sup>	0.5	1.5	<sup>-1.5</sup>
0	1	3	<sup>-3</sup>
1	2	6	<sup>-6</sup>
2	4	12	<sup>-12</sup>

Here is how this data is quickly generated in APL, labeled and plotted.

```
xandys←'x←^{-3+ι5}' 'y1←2*x' 'y2←3*2*x' 'y3←^{-3*2*x}'
ch.Set 'Key' (1↓⊃,/',' ','∘'1↓xandys)
plotxy ⌵xandys
View PG ρ to see it
```



## PLOTTING IN GENERAL IN APL

There is a very extensive plotting library which can do virtually any plot you want. In addition virtually everything can be customized. Fonts and colors can be changed, multiple axes are available, plots can be placed on top of each other, specific areas can be notated or colored etc. To see examples of all of the above and more simply click on each of the following commands.

Samples.Slideshow 3    A Run through selected samples (with 3s delay)  
 ActiveCharts.Active    A Simple illustration of drawing a chart on a form  
 ActiveCharts.Drill    A Sample drill-down application with Dyalog Gui  
 ActiveCharts.Edit    A Sample data editor using draggable markers

## ARE ALL NUMBERS OF FORM ABCABC DIVISIBLE BY 13?

How can that be? Most numbers are not divisible by 13. Lets check it out.

123123÷13

9741                      A yes that one is

264264 813813 547547÷13

20328 62601 42119    A yes those 3 are

Lets write a program to test this out more thoroughly with 3 little fns.

```

rand3u←{ω?9} A make 3 unique random digits a b c with values 1-9
dup2←{10⊔ω,ω} A duplicate a b c and smooshes them together: abcabc
div13←{([x)=x←ω÷13} A x is # ÷ 13. now see if round down ([) of x=x

```

```

1      div13 dup2 rand3u 3      A test it. remember apl works right to left
      A 1 yes the abcabc # is evenly ÷ by 13

```

```

2 5 8      div13 [←dup2 [←rand3u 3 A use output windows to see intermediates
258258      A 3 random digits made by rand3
1           A 3 digits duplicated and smooshed by dup2
           A # ÷ 13 & compare to #'s floor 1=div by 13

```

```

9 5 9 5 3 7      div13" [←dup2" [←rand3u" 3 3 A try it twice using each (")
959959 537537      A the two different a b c's
1 1               A each duplicated & smooshed together
               A each is evenly ÷ by 13

```

```

50000      +/div13" dup2" rand3" 50000p3 A try it 50,000 times & add up 1's(+/)
           A all 50,000 #'s were divisible by 13

```

What if a b and c were not unique numbers? For example is 111111 divisible by 13. Lets revise rand3u to allow non unique numbers and try again.

```

rand3←{?ωp9} A this creates random numbers that may not be unique
rand3" 4p3

```

```

2 1 8 6 1 1 9 6 1 5 8 5 A group 2 and 4 are not unique set if #'s

```

```

50000      +/div13" dup2" rand3" 50000p3 A try with possible non unique a b c's
           A 50,000 non unique are evenly ÷ by 13

```

## RATE WRITING BASED UPON WORD AND SENTENCE LENGTH

First lets store some data in variable lincoln. Here is something he wrote:

If we could first know where we are, and whither we are tending, we could then better judge what to do, and how to do it. We are now far into the fifth year, since a policy was initiated, with the avowed object, and confident promise, of putting an end to slavery agitation. Under the operation of that policy, that agitation has not only, not ceased, but has constantly augmented. In my opinion, it will not cease, until a crisis shall have been reached, and passed. "A house divided against itself cannot stand." I believe this government cannot endure, permanently half slave and half free.



## WHAT IS YOUR NAME WORTH?

If each letter in alphabet was worth a different amount of points (A=1 B=2... Z=26, whose name would be worth the most points?

If A=1 B=2 C=3 . . . Z=26 then ABE would be worth 1+2+5=8 points.

In APL There is an system function `⌈A` which returns the letters in the alphabet.

`⌈A`

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Now we can use dyadic index of(`ι`) to find where in `⌈A` different letters are.

`⌈Aι'ABE'`

1 2 5

⌈ so positions in `⌈A` for ABE are A=1 B=2 E=5

Lets write a fns to get the index numbers of the letters and add them up:

`NAMSUM←{+/⌈Aιω}`

`NAMSUM 'ABE'`

8

⌈ So ABE's score is 8

Lets try on few names:

`NAMES←'JOHN' 'MARY' 'ROBERTA' 'VICTOR' 'TROY'`

`NAMSUM``NAMES`

47 57 79 87 78

⌈ so VICTOR the fourth name wins.

Lets make a labeled table & bar graph so we can see the results better:

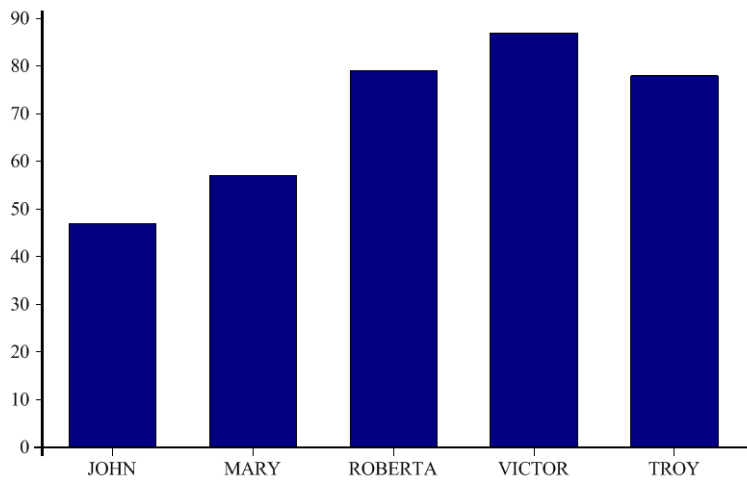
`+DATA← ↑(NAMES)( NAMSUM``NAMES)` ⌈ make table. The + shows the table

JOHN MARY ROBERTA VICTOR TROY

47 57 79 87 78

Not put cursor on DATA and then click on Barchart icon on toolbar at top.

## DATA





## WORKING WITH TABLES

Company wants to compare actual & forecasts for 4 products for 6 months.

```
Forecast←4 6p150 200 100 80 80 80 300 330 360 400 500 520 100 250 350
380 400 450 50 120 220 300 320 350  # Forecast reshape(p) to 4x6 table
```

```
Actual←4 6p141 188 111 87 82 74 321 306 352 403 497 507 118 283 397
424 411 409 43 91 187 306 318 363  # Actual reshape(p) to 4x6 table
```

### Forecast

```
150 200 100 80 80 80
300 330 360 400 500 520
100 250 350 380 400 450
50 120 220 300 320 350
```

### Actual

```
141 188 111 87 82 74
321 306 352 403 497 507
118 283 397 424 411 409
43 91 187 306 318 363
```

### Forecast-Actual

```
9 12 -11 -7 -2 6
-21 24 8 -3 3 13
-18 -33 -47 -44 -11 41
7 29 33 -6 2 -13
```

### Forecast,"Actual

```
150 141 200 188 100 111 80 87 80 82 80 74
300 321 330 306 360 352 400 403 500 497 520 507
100 118 250 283 350 397 380 424 400 411 450 409
50 43 120 91 220 187 300 306 320 318 350 363
```

```
+fa←(c4 0)⌞Forecast,"Actual # each col is 4 wide with 0 decimals
```

```
150 141 200 188 100 111 80 87 80 82 80 74
300 321 330 306 360 352 400 403 500 497 520 507
100 118 250 283 350 397 380 424 400 411 450 409
50 43 120 91 220 187 300 306 320 318 350 363
```

```
(c4 0)⌞Forecast,"Actual,"Forecast-Actual
```

150	141	9	200	188	12	100	111	-11	80	87	-7	80	82	-2	80	74	6
300	321	-21	330	306	24	360	352	8	400	403	-3	500	497	3	520	507	13
100	118	-18	250	283	-33	350	397	-47	380	424	-44	400	411	-11	450	409	41
50	43	7	120	91	29	220	187	33	300	306	-6	320	318	2	350	363	-13

```
((('Prod\Month'),(f''\11tpfa),(f''\11tpfa),'6pc':Fo Act'),fa) # label rows & cols
```

Prod\Month	1:Fo Act	2:Fo Act	3:Fo Act	4:Fo Act	5:Fo Act	6:Fo Act
1	150 141	200 188	100 111	80 87	80 82	80 74
2	300 321	330 306	360 352	400 403	500 497	520 507
3	100 118	250 283	350 397	380 424	400 411	450 409
4	50 43	120 91	220 187	300 306	320 318	350 363

## PLOTTING REGULAR POLYGONS

```

R<-PolyPlot(n s);y;x;y;foot;range;x0;y0;Deg2Rad;theta;i;radius;py;px;pct;area;apothem
  A n is number of sides s=side length. So: PolyPlot 5 10 would plot a 5 sided polygon with
  each side=10
Deg2Rad<={w*01÷180}  A fns to convert degrees to radians for input to trigonometric fns

radius<s÷2×10Deg2Rad 180÷n          A center to a vertex    10 is sine
apothem<s÷2×30Deg2Rad 180÷n         A center to midpt side 30 is tangent
area<(n*s*2)÷4×30Deg2Rad 180÷n      A area of polygon      30 is tangent

x0<y0<0 A x y location of center of polygon on plot
A see http://www.mathopenref.com/polygonregulararea.html for following formulas
theta<(360÷n)×i<0,(1n-1),0 A theta is angle with the x axis plot based on # of sides (n)
x<x0+radius×20Deg2Rad theta+i×(2×01)÷n A x vertice locations 20 is cosine
y<y0+radius×10Deg2Rad theta+i×(2×01)÷n A y vertice locations 10 is sine

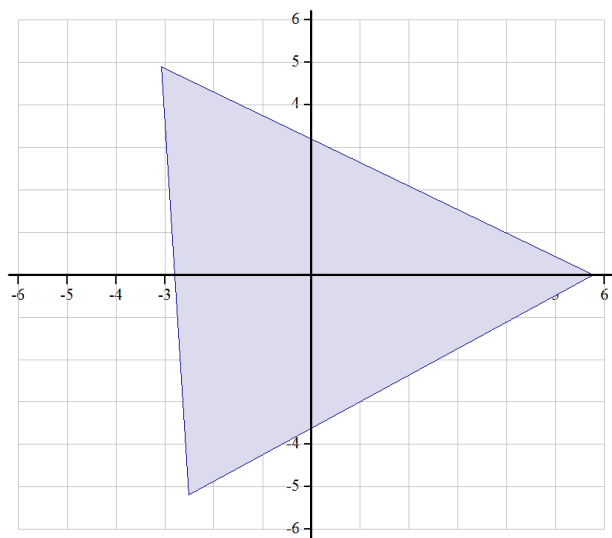
ch.New 350 350 A trying to make x y lengths the same but failing
ch.Set'Head'((n),' Sided Polygon - side length is ',s)
ch.Set'Footer'(('Perimeter=',n*s),(' Radius=',4radius),(' Apothem=',4apothem),('
Area=',4area))
ch.Set''(c''Xrange' 'Yrange'),'range<-1 1×[|x,y
ch.Set''('Xint' 0)('Yint' 0)('forcezero')('XYPLOT,GRID')
ch.Set'style' 'surface'
ch.Plotx y
PG<ch.Close
R<'

```

### PolyPlot 3 10

View PG 9 to see it

*3 Sided Polygon - side length is 10*

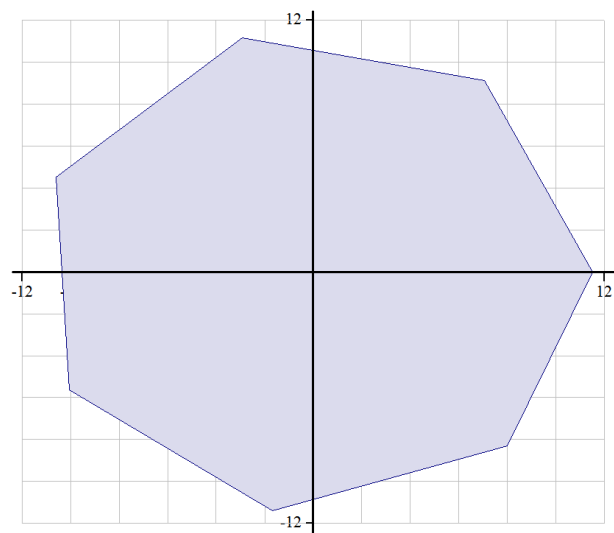


Perimeter=30 Radius= 5.7735 Apothem= 2.8868 Area= 43.3013

### PolyPlot 7 10

View PG 9 to see it

*7 Sided Polygon - side length is 10*



Perimeter=70 Radius= 11.5238 Apothem= 10.3826 Area= 363.3912

## APL REFERENCES

For noncommercial use you can get a free version of this APL at: <http://dyalog.com/> This includes everything. There are thousands of pages of online manuals and tutorials describing everything available.

[http://en.wikipedia.org/wiki/APL\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/APL_(programming_language)) A Programming Language (APL)

Here is a video showing how to do the Game of Life in APL. This video will give you an idea of the amazing power and conciseness of APL but is probably beyond your ability to really understand. <http://www.youtube.com/watch?fmt=18&gl=GB&hl=en-GB&v=a9xAKttWgP4>

Some educational videos at: <http://www.youtube.com/user/APLtrainer>

[  
Complete free APL tutorial at: <http://aplwiki.com/LearnApl/LearningApl>

Another extensive(800+ pages) tutorial you can download for free <http://www.dyalog.com/MasteringDyalogAPL/MasteringDyalogAPL.pdf>

Another apl tutorial with a sandbox where you can try out lines of APL code such as from this tutorial except for the plotting things. [www.tryapl.org](http://www.tryapl.org)

Some information about Kenneth E. Iverson the inventor of APL. He was a Harvard mathematics Professor, worked for IBM and won a Turing Award for creating APL. He first developed APL as a concise notation for mathematics. Later he developed it as a computer language. [http://en.wikipedia.org/wiki/Kenneth\\_E.\\_Iverson](http://en.wikipedia.org/wiki/Kenneth_E._Iverson)