```
using System;
using System.Collections.Generic;
using System.Text;
using APLNext;

namespace VAPL_MEAN{
    public class MEAN{
        public function double ARITHMETIC_MEAN(DV){
            return (+/DV)÷⍴DV
        }
        public function double GEOMETRIC_MEAN(DV){
            return (×/DV)*÷⍴DV
        }
        public function double[] MODE(DV){
            X←,+/[0]<\DV∘.≈DV
            X←(X≈⌈/X)/DV
            return X
        }
        public function double RMS_MEAN(double[] DV){
            X←+/DV*2
            X←X÷⍴DV
            X←X*0.5
            return X
        }
        public function double MID_RANGE(double[] DV){
            X←⌊/DV
            X←X+⌈/DV
            X←X÷2
            return X
        }
        public function double MEDIAN(double[] DV){
            X←⌊.5×¯1 0+⍴DV
            X←.5×+/DV[X]
            return X
        }
```

VisualAPL - APL in the programming mainstream

This is a simple VisualAPL script illustrating some familiar statistical functions. Observe how the APL language eliminates superfluous programming structures to clearly provide the desired functionality. This code snippet can be run as a script or it can also be incorporated into a .Net assmebly and compiled. The resulting .Net assembly can be consumed by any .Net language. As a .Net assembly it can also be exposed as COM to legacy Win32 software.

For more information, contact: sales@apl2000.com

Joe Blaze 20061107
These functions assume the index origin (⎕io) is 0. These functions have been presented as multi-line functions for clarity. The listing below incorporates comments which may be useful for new APL'ers:

```
namespace VAPL_MEAN{
    public class MEAN{
        public function double ARITHMETIC_MEAN(DV){
            return (+/DV)÷⍴DV
//The reduction operator (/) applies the plus function (+) to the elements of DV
//The VisualAPL shape function (⍴) returns the number of elements of DV
        }
        public function double GEOMETRIC_MEAN(DV){
            return (×/DV)*÷⍴DV
//The reduction operator (/) applies the multiplication function (×) to the elements of DV
//The VisualAPL shape function (⍴) returns the number of elements of in DV
//The monadic divide function (÷) returns the multiplicative reciprocal of (⍴DV)
//The dyadic exponentiation function (*) returns (×/DV) raised to the power ÷⍴DV
        }
        public function double[] MODE(DV){
            X←,+/[0]<\DV∘.≈DV
//Obtain a numeric vector indicating the frequency of each value in DV
            X←(X≈⌈/X)/DV
//Obtain the value(s) which occur most frequently in DV
```

```
            return X
        }
        public function double RMS_MEAN(double[] DV){
            X←+/DV*2
//Square all elements of DV and then obtain the total of these values
            X←X÷⍴DV
//Divide the total by the number of elements in DV
            X←X*0.5
//Obtain the square root of the result
            return X
        }
        public function double MID_RANGE(double[] DV){
            X←⌊/DV
//Obtain the value of the smallest element of DV
            X←X+⌈/DV
//Add to it the value of the largest element of DV
            X←X÷2
//Divide the result by 2
            return X
//Return the arithmetic mean of the largest and smallest values of DV
        }
        public function double MEDIAN(double[] DV){
           X←⌊.5×¯1 0+⍴DV
//Find the index in DV of the “middle” element(s) of DV
          X←.5×+/DV[X]
//Obtain arithmetic mean of the “middle” element(s) of DV
           return X
        }
```