GSD 6349 Mapping II : Geosimulation
Professor Robert Gerard Pietrusko
<rpietrusko@gsd.harvard.edu>
(c) Fall 2016

Please cite author and course when using this library in any personal or professional project.

class Lattice()

Lattice() provides numerous methods for storing data in and retrieving data from a two-dimension grid– for use with cellular automata and agent-based models designed in Processing.

```
      ///////////////
     //           //
    //  METHODS  //
   //           //
  ///////////////
```

```
Lattice(int _w, int _h)
Lattice( int _w, int _h, float val )
Lattice(int _w, int _h, float _min, float _max )
Lattice(int _w, int _h, float _min, float _max, String _round )
Lattice(int _w, int _h, int _min, int _max )
Lattice(int _w, int _h, int _min, int _max, float prob )
```

```
void        replaceWith( Lattice l )
void        put( int x, int y, float val )
void        put( PVector pv )
float       get( int x, int y )
float       getNorm( int x, int y)
PVector     getcell( int x, int y)
PImage      getPImage()
float[][]   getlattice()
void        lock( int x, int y)
void        unlock( int x, int y)
void        lockAll()
void        unlockAll()
float       max()
float       min()
float       average()
PVector[]   histogram()
```

```
     ////////////////////
    //  CONSTRUCTORS  //
   ////////////////////

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

      Lattice(int _w, int _h)

            INPUTS
                  int w the  width of the lattice (columns)
                  int h the height of the lattice (rows)
            creates a w x h lattice initialized with the value 0.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

      Lattice( int _w, int _h, float val )

            INPUTS
                  int   w   the  width of the lattice (columns)
                  int   h   the height of the lattice (rows)
                  float val desired initial value
            creates a w x h lattice initialized with the value val.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

      Lattice(int _w, int _h, float _min, float _max )

            INPUTS
                  int   w   the  width of the lattice (columns)
                  int   h   the height of the lattice (rows)
                  float min minimum random value of lattice
                  float max maximum random value of lattice
            creates a w x h lattice populated with random values [min,max]

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

      Lattice(int _w, int _h, float _min, float _max, String _round )

            INPUTS
                  int   w      width  of the lattice (columns)
                  int   h      height of the lattice (rows)
                  float min    minimum random value of lattice
                  float max    maximum random value of lattice
                  String round enter the word "ROUND"

      when the word "ROUND" is entered as the last parameter, a lattice
      of size w x h is created with random INTEGER values between [min,max]
```

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

      Lattice(int _w, int _h, int _min, int _max )

            INPUTS
                  int   w   the  width of the lattice (columns)
                  int   h   the height of the lattice (rows)
                  int   min minimum random dichotomous value of lattice
                  int   max maximum random dichotomous value of lattice
      when min and max are forced to be INTs, it creates a WxH sized lattice
      populated with random values that are EITHER min OR max

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

      Lattice(int _w, int _h, int _min, int _max, float prob )

            INPUTS
                  int   w    the  width of the lattice (columns)
                  int   h    the height of the lattice (rows)
                  int   min  minimum random dichotomous value of lattice
                  int   max  maximum random dichotomous value of lattice
                  float prob the probability of a cell being value 'max'

      when min and max are forced to be INTs, by using this constructor, it
      creates a WxH sized lattice populated with random values that are
      EITHER min OR max. where max is generated with a probability of 'prob'
      and min is generated with a probability of '1-prob'.
```

```
    ////////////////////
    //  PUT METHODS   //
   ////////////////////

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

void        replaceWith( Lattice l )

            replaces the current lattice with the Lattice l of the
            input. if the incoming lattice is of a different size the current
            one, W and H are updated to match the new size.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

void        put( int x, int y, float val )

            INPUTS
                  int x the x coordinate of the cell being updated
                  int y the y coordinate of the cell being updated
                  float val the value being put into cell x,y

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

void        put( PVector pv )

            INPUTS
               PVector pv stores value pv.z at cell location pv.x, pv.y
    ////////////////////
    //  GET METHODS   //
   ////////////////////

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

float       get( int x, int y )

            INPUTS
                  int x the x-coordinate of the cell being accessed
                  int y the y-coordinate of the cell being accessed
            OUTPUT
                  returns the value of the cell at location x,y

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

float       getNorm( int x, int y)

            INPUTS
                  int x the x-coordinate of the cell being accessed
                  int y the y-coordinate of the cell being accessed
            OUTPUT
returns the normalized value of the cell at location x,y scaled between 0.0
and 1.0 where 0.0 corresponds to the minimum value of the lattice and 1.0
corresponds to the maximum value of the lattice

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

PVector     getcell( int x, int y)
            INPUTS
                  int x the x-coordinate of the cell being accessed
                  int y the y-coordinate of the cell being accessed
            OUTPUT
                  returns a PVector with the following:
                  PVector.x the x-coordinate of the cell being accessed
                  PVector.y the y-coordinate of the cell being accessed
                  PVector.z the value of the cell located at x,y

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

PImage      getPImage()

Returns an image of the entire lattice with the minmum value mapped to the
color 0,0,0 and the maximum value mapped to the color 255,255,255

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
float[][]   getlattice()

Returns the lattice as a 2D float Array.
```

```
     //////////////////////////
    //  SET ACCESS METHODS  //
   //////////////////////////

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

void  lock( int x, int y)

      INPUTS
            int x the x-coordinate of the cell being locked
            int y the y-coordinate of the cell being locked

      simply locks a cell so that its value cannot be modified
      by put methods.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

void  unlock( int x, int y)

      INPUTS
            int x the x-coordinate of the cell being unlocked
            int y the y-coordinate of the cell being unlocked

      unlocks a previous locked cell so that its value can
      again be modified by put methods.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

void  lockAll()

      locks all values of a lattice so that none can be modified
      it is a read-only operation.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

void  unlockAll()

      unlocks all cells.
     //////////////////////////
    // BASIC LATTICE STATS  //
   //////////////////////////

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

float  max()

       returns the maximum value stored in the Lattice.

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

float  min()

       returns the mimimum value stored in the lattice

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

float  average()

       returns the average value stored in the lattice

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

PVector[]   histogram()

            creates a histogram of the values in the Lattice.
            returns an arry of PVectors with the following form:
            PVector.x : Value in the Lattice
            PVector.y : Number of occurances of that value

WARNING: this method is intended for integer / nominal data. Technically it
will work with floats but it will be heavy and not very helpful; You will
likely get a histogram bin for every single lattice cell. USE WITH CARE
```