Usage

./gistpp –i infile –op operation [-i2 infile2][-o outfile][-opt options]

Table 1. List of operations with a brief explanation of their function.

|  |  |
| --- | --- |
| Operation | Function |
| group | Groups a binary dx file based on spatial proximity |
| sas | Creates a dx map of the real solvent accessible surface, which comprises voxels where the water density is 0.1 of bulk density and has a neighboring voxel with less than 0.1 bulk density. |
| mult | Multiply two dx files together |
| add | Add two dx files together |
| sub | Subtract dx file 2 from dx file 1 |
| div | Divide dx file 1 by dx file 2 |
| addconst | Add a value to each voxel in a dx file |
| multconst | Multiply a value to each voxel in a dx file |
| filter1 | Process a dx file for whether it fits a single criteria within its own voxels |
| sum | Integrates over the provided dx map printing the sum |
| defbp | Masks a volume around a pdb structure by flagging all voxels within a user defined distance of all heavy atoms. |
| printpdb | Converts a dx file to pdb format. The output pdb file will contain voxel coordinates as hydrogen atoms in which the occupancy column corresponds to the original voxel data. |
| Makedx | Makes a dx map from a column in the space delimited text output of GIST |

Each of these functions will be explained in greater detail below.

For help gistpp allows users to request further information regarding operations and options via:

./gistpp –operations

./gistpp –options

and full help via:

./gistpp –h (or simply ./gistpp)

**Group**

Example usage:

./gistpp -i infile.dx –op group

The grouping function will read in the desired dx file and begin creating groups around voxels that contain non-zero values by searching all neighboring voxels recursively. This is done until no new group members are found (meaning all surrounding voxels are zero) or the heap is overloaded due to high group occupancy. To bypass heap overload groups cannot exceed occupancy of 10000.

Typical input files to this operation are dx files which have been previously processed and are in binary format (1 for data, 0 for empty) or a dx file type which contains 0 as a base case such as the g(X) file types.

The output of this function will be several files. A groupNN.dx file for each NN group that contains a dx file that only contains non-zero values in voxels that are within that group (the values stored will be the same voxel values that are present in the original input file). A grcount.txt file that lists each group by number and states their voxel occupancy. A mastergroup.dx and mastergroup.pdb file which contain location data for every group created during this process.

The benefit of the pdb output is the ability to visualize the file and identify groups of interest based on proximity to some feature of interest in the structure analyzed through gist. Specifically group number information is stored in the column within the pdb format reserved for resid.

**SAS**

Example usage:

./gistpp -i gist-gO.dx –op sasa -o SASA.dx

The SASA function must be run on a g distribution (typically desired to be used on oxygen distributions). This code searches through the dx file voxel by voxel until it finds a voxel with greater than 0.3. This value corresponds to an approximate cutoff representing water is present in this voxel during the simulation in question. Then each neighbor of that voxel is checked, if even one neighbor contains a value less than 0.1 the original voxel is flagged as being a part of this surface. The value 0.1 represents an approximate cutoff that represents a voxel that does not contain water throughout the simulation. Once all voxels have either been flagged as part of the surface or excluded a binary dx file is printed in which only flagged voxels contain values. This produced dx file when visualized will show the surface which is accessible to the solvent during the course of the simulation which is being analyzed.

**Mult**

Example usage:

./gistpp -i infile.dx -i2 infile2.dx -op mult -o outfile.dx

This operation will multiply file1 and file2 together. Producing a dx map where each voxel is the product of the corresponding voxel in each input file.

**Add**

Example usage:

./gistpp -i infile.dx -i2 infile2.dx -op add -o outfile.dx

This operation will add file1 and file2 together. Producing a dx map where each voxel is the sum of the corresponding voxel in each input file.

**Sub**

Example usage:

./gistpp -i infile.dx -i2 infile2.dx -op sub -o outfile.dx

This operation will subtract file2 from file1. Producing a dx map where each voxel is the difference of the corresponding voxel in each input file.

**Div**

Example usage:

./gistpp -i infile.dx -i2 infile2.dx -op div -o outfile.dx

This operation will divide file1 by file2. Producing a dx map where each voxel is the quotient of the corresponding voxel in each input file.

**Addbyconst**

Example usage:

./gistpp -i infile.dx -op addconst–opt const C -o outfile.dx

This operation will produce a new dx map where each voxel value is the sum of the corresponding voxel in infile and the user defined constant C.

**Multbyconst**

Example usage:

./gistpp -i infile.dx -op multconst –opt const C -o outfile.dx

This operation will produce a new dx map where each voxel value is the product of the corresponding voxel in infile and the user defined constant C.

**filter**

Example usage:

./gistpp -i infile.dx –op filter1 –opt cutoff1 X [-opt gt or lt] -o outfile.dx

This operation will return a binary dx file where voxels will be flagged as 1 if the corresponding voxel in infile 1 meets a defined criterion and 0 if the corresponding voxel does not. The criterion is defined by the options: cutoff1 and gt1 or lt1. These options define the threshold value (cutoff1) and whether the desired quantities are higher than or less than said threshold (gt for

**Sum**

Example usage:

./gistpp -i infile.dx –op sum

This function will read in all voxel values in the infile and sum them. Then print that value along with the average per voxel quantity in the command line.

**Defbp**

Example usage:

./gistpp –i infile.dx –i2 ligand.pdb –op defbp –opt const X –o outfile.dx

\*Note the pdb format file must be entered in i2 NOT in i.

This operation defines a volume around an input pdb file by flagging all voxels within a distance X of all heavy atoms within the pdb file and outputting a binary dx map of those flagged voxels.

This operation can be used to find a binding pocket (where it derives its name) or in combination with the sum and mult operations to integrate any values around a heavy atom of interest. The workflow would be to define the region with defbp, mult the resulting output with a dx file containing values, then sum the final output.

**Printpdb**

Example usage:

./gistpp –i infile.dx –op printpdb

This operation will read in the dx file and print a pdb file in which each voxel from the dx file will be a hydrogen atom with an occupancy value related to the original dx file voxel value.

**Makedx**

Example usage:

./gistpp –i infile.out –i2 infile.dx –op makedx –opt const ##

This operation outputs the data found in column ## from infile.out into a dx file with the same header information as infile.dx. Both infile’s must be from the same gist output. This operation allows end users to create visualizations of data not output by the default gist run, such as normalized quantities of energy and entropy.

Example output of ./gistpp –operation:

Following is a list of the available operations provided in this code

The format is as follows:

-op [operation][# infile][type infile][outfile][synopsis]

where:

operation is the command flag

#infile = number of infiles required (1 or 2)

type infile is the required infile format(dx, gist, or pdb where: gist is the gist outfile)

outfile is whether or not the code requires a specified outfile (Y/N)

synopsis is a brief explanation of the operation

-op group 1 dx N produces files which contain spatially grouped voxels with similar qualities

-op contour 2 dx Y produces an outfile which contains the values of the first provided the same voxel meets a criteria in the second file

-op clear1 1 dx Y produces an outfile which contains all 1's and 0's, 0 if the voxel does not fit a desired criteria in infile (ex: high energy), 1 if the voxel does

-op clear2 2 dx Y same as clear1 only requires the voxel to fit criteria in 2 dx files rather than simply 1 (ex: high energy and high g(O))

-op sasa 1 dx Y produces a dx file which contains 1's in each voxel which defines the solvent accessible surface area

-op sum 1 dx N prints in the command line the sum of all voxel quantities and the average voxel quanitity for the values found in the dx file provided

-op add 2 dx Y produces a dx file in which each voxel is the sum of the 2 corresponding voxels in each input file

-op sub 2 dx Y produces a dx file in which each voxel is the difference of the 2 corresponding voxels in each input file (file 1 - file 2)

-op div 2 dx Y produces a dx file in which each voxel is the quotient of the 2 corresponding voxels in each input file (file1/file2)

-op mult 2 dx Y produces a dx file in which each voxel is the product of the 2 corresponding voxels in each input file

-op addconst 1 dx Y produces a dx file in which each voxel is changed by the addition of a specified constant

-op multconst 1 dx Y produces a dx file in which each voxel is changed by the multiplication of a specified constant

-op defbp 2 dx+pdb Y produces a dx file in which each voxel is flagged as 1 when within a set distance of any ligand.pdb heavy atoms

-op printpdb 1 dx Y produces a pdb file containing hydrogen atoms at every voxel with the gist voxel data stored in occupancy

-op popstat 1 gist N prints in the command line statistics taken directly from the gist outfile

Following is an example of ./gistpp –options:

Following is a list of the available options and which operations they correspond to

The format is as follows:

-opt [option][##][operation][synopsis]

where:

option is the command flag

## indicates a numerical value must be added after the option

operation is the list of operations this option is required for

synopsis is a brief summary of the options effect

-opt cutoff1 ## [clear1 clear2 defbp] this option specifies the cutoff value which is desired to be applied to infile or the desired distance in defbp

-opt cutoff2 ## [clear2 contour] this option specifies the cutoff value which is desired to be applied to infile2

-opt gt1 [clear1 clear2] this option specifies we are interested in values greater than cutoff1 in infile

-opt lt1 [clear1 clear2] this option specifies we are interested in values less than cutoff1 in infile

-opt gt2 [clear2 contour] this option specifies we are interested in values greater than cutoff2 in infile2

-opt lt2 [clear2 contour] this option specifies we are interested in values less than cutoff2 in infile2

-opt const ## [addconst multconst] this option specifies the constant to add or multiply by in the provided function