Calculate g(r)

You can calculate the second order radial distribution function, g(r), using gr.c. Compile gr.c by invoking your c compiler on gr.c. To do this:

1. Open your terminal app
2. Navigate to the directory that gr.c is in
3. Figure out the name of your c compiler program (google this)
4. Typically it will be called either gcc or cc, if it goes by a different name then replace gcc with that name in the following
5. Run: gcc -O3 gr.c -o gr
    1. -O3 (note that it is the capital letter O) turns on the highest levels of optimization by the compiler
    2. -o gr specifies that we want the binary to be named gr
        1. replace gr with whatever you want the binary to be named

6. Then, you can run the g(r) program by typing ./gr (arguments)
I recommend adding the directory gr is in to your path (just google this)

The arguments to gr.c are as follows (all are required)
input file -> a file in .tr format
output file name -> the name of the file that you want to store the (x,y) pairs of the g(r) plot in
diameter -> the diameter of the smaller particle, or whatever particle size you want to correspond to x = 1 on the g(r) plot
So, for example you could do:
gr dump.tr outplot.dat 2.56 to run gr on the track file in dump.tr and store the results in outplot.dat with x = 1 corresponding to a diameter of 2.56

Note it will overwrite whatever was stored in outplot.dat

Convert .xyz to .tr

You can convert lammps .xyz files to track files using the read_tr.py script. To run it, make sure that you have python installed (it's tested with 2.7 but should work with others) and also have numpy installed. Then, run:
python read_tr.py (name of .xyz file) (name of track file you want to create) and it will save the track file in the specified location

Note that it will overwrite anything stored in (name of track file you want to create)

Calculate s(k)
You can calculate the 2D s(k) and it's related functions using sk.py. To do so, open the python REPL, then do the following:

```
>>> import sk
>>> sofk = sk.run(track file to run it on
>>> skavg = sk.runskAverage(sofk)
```

Calling sk.plotSKAverage(skavg) or sk.plotSK(sofk) will produce the plots of the data in sofk and akavg

Misc. useful scripts:
Get all tiffs from the cluster
Run: getTiffs (file name) where file name is the directory you want to put the tiffs in

Find differences in files in a more readable manner
Run: diffD (file 1) (file 2)

Plot from the command line (requires gnuplot)
Run: plot (file to plot) (optional arguments)
If no arguments are provided it will plot to the screen with lines connecting points
If it's called as plot (file to plot) -nl then it will plot without lines connection points
If it's called as plot (file to plot) -l -o (output file) then it will save the plot to the output file with lines connecting points

Todo:
This can certainly be made better through Latexing it or just using better formatting
Add s(k) theta dependence part (it wasn't up when I made this)
Have someone who hasn't used the programs read over this to see if it actually made sense