Using this document:

Linux shell commands are preceded by **$** while commands in the **cpptraj** interface are preceded by **>**. Do not type these characters when entering the commands.

Commands in general are formatted in **bold monospaced font**. Variables are in regular monospaced font.

Text that you should replace is surrounded by angled brackets **<replace this text>**.

Optional commands are surrounded by square brackets **[optional_argument]**.

Instantaneous results and averages for plotting:

Once you're done you will want to analyze and collect average temperature, pressure, data. Amber provides a handy perl script that will do a lot of this collection for you. First, make sure you have Amber loaded:

```
$    module load amber
```

On Expanse, load Amber with

```
$    module load cpu/0.15.4 gcc/9.2.0 openmpi/3.1.6 amber/20
```

Next, run the perl script as follows:

```
$    process_mdout.perl <output file>
```

A brief review of cpptraj analysis for water and salt solution simulations.

Cpptraj is a useful postprocessing tool for our trajectories (it is a speedier version of ptraj). It allows you to do a wide variety of things including calculating geometric properties on a residue-by-residue or atom-by-atom basis in time. It also lets you average, window, and cluster your trajectories. For the homework analysis for water simulations, we will use a few of the many commands cpptraj supports. The full AMBER manual and cpptraj manual is available in a single document: http://ambermd.org/doc12/Amber20.pdf

This document is hundreds of pages long, so we've provided the very basics of commands that you'll need just for accomplishing tasks in the homework. Make sure Amber is loaded (see above).

For more in depth option information on these commands, you can look at AMBER-hub, but it lacks some explanation.

In all cases we can start **cpptraj** with the following command:

```
$    cpptraj -p <prmtopfile>
```

Then load up the trajectory you want to analyze:

```
>    trajin <mdcrd file>
```

Note: If you just wanted to visualize the trajectory and make sure that everything is neatly centered you can use the **autoimage** command.

```
>    autoimage
```

This command centers around a solute molecule or the first residue if there does not appear to be a solute and then wrap every molecule into the box. For now, we don't need to do this. Instead we're interested in carrying out analysis.

Generally, we first set up a series of desired commands within **cpptraj** and then run them all together.

Note: These are meant to be typed at the prompt you see after starting **`cpptraj`**, not at the normal Linux commandline.

Radial distribution functions

`radial <outfilename> <spacing of points in RDF> <maximum value for RDF> <solvent mask 1> [<solvent mask 2>] [density <density>] [intrdf <file>] [rawrdf <file>]`

There are a few other options that the radial command takes, but these are the most relevant.
-The `outfilename` is where the RDF is stored.
-Spacing of the points should be e.g. 0.1Å or less
-Maximum value shouldn't be more than ½ the size of the cell (e.g. 15 Å).

-`solvent mask` – masks in Amber are as follows, for atom name we would use `@O` or `@MG`, for residues we would use `:1-2` to represent residues 1 and 2. There are more complicated combinations explained in the Amber manual. For O-O RDF we only put one `@O` (solvent mask #2 is optional here) but we need to specify `@MG` for an Mg-O RDF.
-`density` – the default here is 0.033456 mols/Å^3. This is pretty close to the density you'll find in pure water simulations, but you need to calculate this explicitly for the Mg-O case (use the average box size from your simulation).
-`intrdf` specifies a file that gives the integrated RDF.
-`rawrdf` gives the unnormalized RDF.

Diffusion coefficients

`diffusion out <name of diffusion file> time <time_per_frame> <filename prefix> <mask> [average]`

-time per frame: this is how frequently mdcrd gets written in ps, can figure out from **`dt*ntwx`** in your MD input files (**Note:** remember to do this math!).
-`filename prefix`, can leave this blank and will default to 'diffusion'
-`mask` – `:*` will be all atoms, which is what we want here.
-`average` specifies using an average and not just printing instantaneous contribution from each molecule.

The Einstein relation states that the diffusion coefficient is related to mean square displacements. To obtain them from the output, use overall mean square displacements in Å^2 (diffusion_r.xmgr). Then obtain the slope via linear regression and multiply by 10.0/6.0. The resulting value is the diffusion coefficient in units of 1x10^{-5} cm^2/s (converted from Å^2/ps).

Distances

`distance <name of distance> <mask 1> <mask 2> out <filename>`

Different distances can be written as multiple columns to one filename. You want to specify the masks of the two objects you are interested in the distance of. For our ions for instance this would be :1 vs :2 or :1 vs :3.

Water shell properties

`watershell <mask> <filename> [lower <lower_bound>] [upper <upper_bound>] <solventmask>`

- `mask` – The mask of the solute ion with water shell calculations
- `filename` – the name of the output file in which to write
- `lower_bound` – the cutoff for which to consider the "first" coordination shell
- `upper_bound` – the cutoff for which to consider the "second" coordination shell
  The defaults are 3.4 and 5.0 Å for these two bounds – you will want to change it based on the RDFs (i.e. where the local minima occur after the first local maxima)
- `solventmask` – can specify which atoms correspond to the solvent.

-*Note*, if your mask captures multiple copies of the same species, the number of water molecules reported in a shell will be the total for all copies. If I had 3 Na^+, my total number of water molecules in my first shell would be all the water molecules in the first shell of each Na^+.

Hydrogen bonding information

```
hbond [out <filename>] [<mask>] [donormask <dmask>] [acceptormask
 <amask>] [avgout <filename>] [solventdonor <sdmask>]
 [solventacceptor <samask>] [image]
```

Calculates hydrogen bonds for "solutes". In lab, we treat a single water molecule as the "solute" and the rest as "solvent".

- mask – Atoms to treat as solute.
- `image` – perhaps the most important option, this tells it there are periodic boundary conditions.

Closest residues to another residue **(e.g. water to Mg or Cl)**

```
closest <# to keep> <mask of centered residue> closestout
<filename>
```

-Keep a fixed number of water residues around a centered residue, reports the distance of the two, gives the original solvent molecule number and frame #.

Position of the center of mass

```
vector center out <filename> <mask1> <mask2> magnitude
```

This will measure the vector position of the center of mass of atoms in <mask 1> and <mask 2>, as well as the distance between them, and write the output in columns to <filename>. You can use * in for <mask 1> to select all atoms, and if you leave out <mask 2>, it will be taken as the origin. The magnitude keyword is optional and leaving out will skip printing the distance between the masked vectors.

For all cpptraj analysis you must issue your commands and then type:

```
>     go
```

To execute the commands.

Note: You'll know they're running when you see progress statements (eg. the trajectory being read in and processed). You can combine multiple commands or run each time to carry out a separate command.

All cpptraj commands can also be written to a file and then fed into cpptraj, e.g.:

```
$     cpptraj -p <prmtopfile> -i cpptrajcommandfile
```