Now that we have a txt file, we need to generate gro and itp files. But before that there are intermediate steps.

Task A. Given Smiles of a molecule, return the x, y, and z coordinate of each atom, indexed by the appearance of the atom by tracing the smiles code.

note:

RDKit gives consistent coordinated while OpenBabel gives seemingly random coordinates

when calculate the distance between index 0 and 1 and 1 and 2, here are the results:

RDKit gives 1.50 and 1.22

Openbabel over 3 separate runs on the same molecule gives:

1.51 and 1.22

1.55 and 1.23

1.49 and 1.23

-- it seems like that the position of the atoms can change but the distance between the atoms seems to be approximately the same, I will use RDKit for consistency

Task B. Calculate Bead relative position and append them to the text file

We already have the main function that create the text file called

def create_beads_text_file(final, mapping, compound_name, smiles):

Now we need to add 3 more columns called x,y,z that calculate the relative position of the beads.

We can do this by first calling the function get_coordinates_from_smiles

to get coordinate of each atom in the form that looks like [{'index': 0, 'symbol': 'C', 'x': -7.026230834556532, 'y': -5.136800761230697, 'z': -1.8775043431846494}, {'index': 1, 'symbol': 'C', 'x': -7.403909080437602, 'y': -3.706023660564536, 'z': -2.142130058489784}, ...

now that we have this, we can figure out the x, y, z of each bead by calculating the average location of all of the atoms of each bead based on the index

Might as well add mass into the text file.

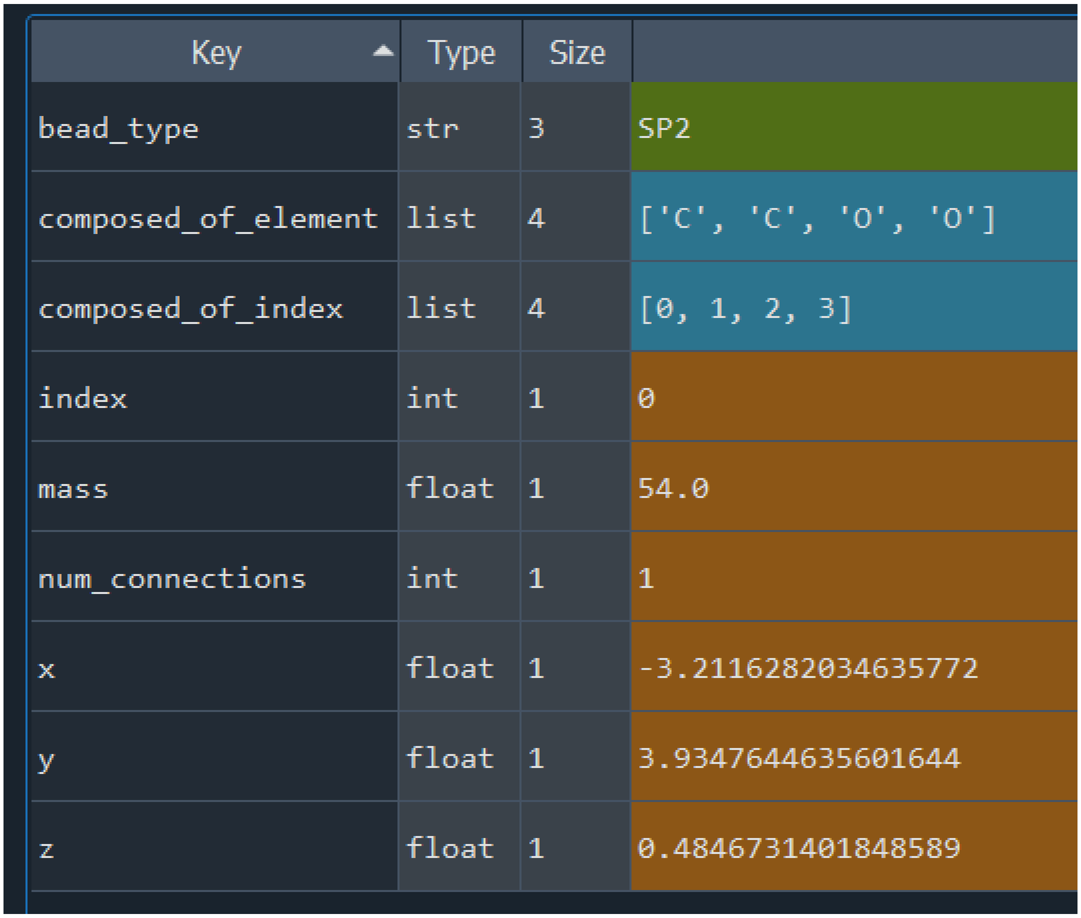
The rule for adding mass is 36 for T, 54 for S, and 72 for the large beads

Task C: We do not want the text file anymore, now what we want is two intermediate output so that we can make the itp and the gro files

instead of the text files, we now can just make 2 separate output files, 1 is for each beads and 1 is for all of the connections

instead of creating this, can we have 2 outputs that is the dictionary for the top half and the dictionary for the bottom half of the text files. so the first output will be something like [{index:0 etc… and the second output will be something like [{index:0, connection_index….

After this step, we will have inputs that looks like



| Key | Type | Size | |
|---|---|---|---|
| bead_type | str | 3 | SP2 |
| composed_of_element | list | 4 | ['C', 'C', 'O', 'O'] |
| composed_of_index | list | 4 | [0, 1, 2, 3] |
| index | int | 1 | 0 |
| mass | float | 1 | 54.0 |
| num_connections | int | 1 | 1 |
| x | float | 1 | -3.2116282034635772 |
| y | float | 1 | 3.9347644635601644 |
| z | float | 1 | 0.4846731401848589 |

now we will try to format for the next step

First, gro files, this file only needs

Line 1: a title (any text, up to 80 chars)

Line 2: the number of atoms (integer)

Lines 3→N+2: one atom per line

Last line: box vectors (3 or 9 numbers), each formatted as %10.5f, e.g.:  5.00000  5.00000  5.00000

gro_maker(beads_data, compound_name)

we only need beads_data and compound_name for this.

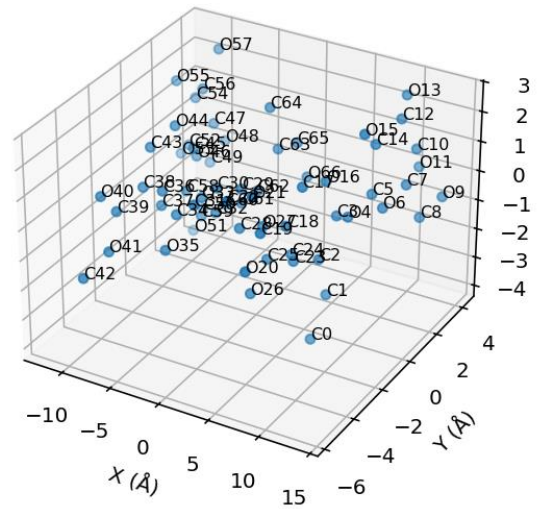
we have all the information that we need, all we have to do now is the formatting, let the first line and each atom have the 1+compound_name

Second, itp files, this file needs a few more things…

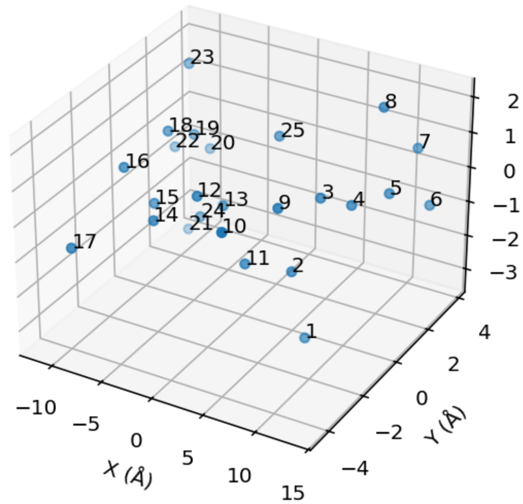
Next, I have to fix the batch running files so that it can run TPCN and Cancer properly

now for visualization we have TPCN00004:

3D Structure from SMILES Coordinates



3D Structure from BEADS Coordinates



next I looked at the new paper and is able to create a python code that let me create a csv file from the pdf file.

ran the code, the result shows:

353 works properly

300 does not work

no fake working run -> seems like with simpler task it can only either works or not works

Note:

Most cases that do not work is because we do not have a bead that represents it

-> Shows error logs

I download the code and install the necessaries and after debugging (my computer) I was able to run the Github auto_martini and everytime I run It gives me a itp file of the beads that it uses.

```
python -m auto_martini --smi "{SMILES}" --mol NAME --top NAME.itp
```

```
python -m auto_martini --smi "c1ccccc1Cl " --mol NAME --top NAME.itp - this example takes 10 seconds
```

Seems like lots of testing is needed to be done here

I have also added more logs so I know exactly which bead is missing and fix an oversight from the 5-ring bead mapping in which it assumes something which turns out to be only true most of the time.

After this and a rerun the result is only 302 correct and 351 fails

but these correct results are much more guaranteed

Seems like this data set have 1 thing that is detrimental to my code and will probably needs fixing, it has many different atoms inside the ring such as I, Br, S, F which I have not accounted for ever because I have never seen it. however I was able to keep a ~50% working accuracy, consistent with TPCN

We have half the errors being “non-benzene 6-ring section has non C/O atom”

and the other half are just beads that we do not have:

Extra notes:

I have some suspicions about using this paper as a helper to our paper.

The martini 2 model seems to be stuck in an infinite loop as more big atoms are added -> any example will take at least ~ 5 seconds while taking a small example of TPCN with 24 atoms, the algorithm never execute.

Note: they handpicked these specific examples that work for their models.

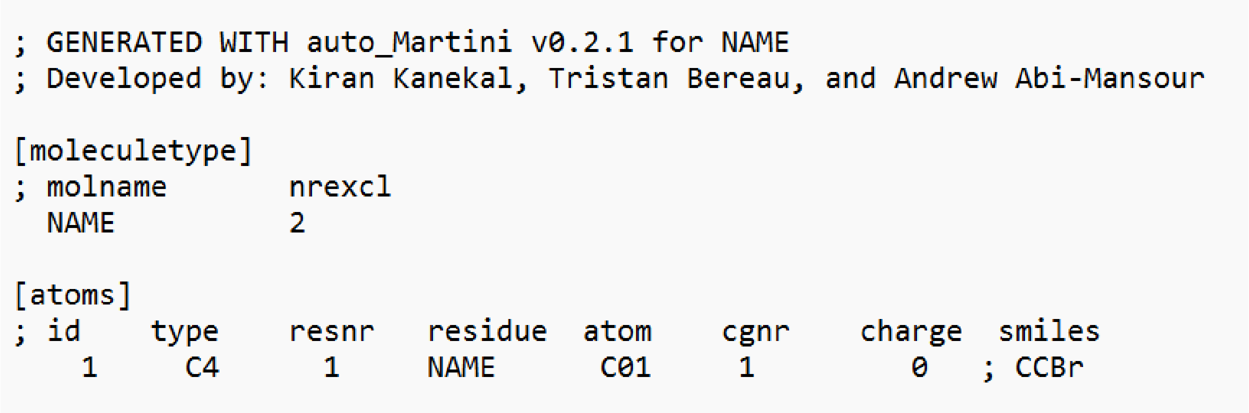
Example: c1ccccc1Cl works

but even if I add an oxygen like Oc1ccccc1Cl it stop working -> very limited in what it can do

The beads they give are very different from the building blocks -> we can use any example here and it will show

P1 -> C3

SX2 -> C4 -> this example is just CCBr...



```
; GENERATED WITH auto_Martini v0.2.1 for NAME
; Developed by: Kiran Kanekal, Tristan Bereau, and Andrew Abi-Mansour

[moleculetype]
; molname       nrexcl
  NAME          2

[atoms]
; id    type    resnr   residue  atom    cgnr    charge  smiles
    1     C4      1     NAME      C01     1          0   ; CCBr
```

*I also have a suspicion that Martini 2 and Martini 3 are way too different*.