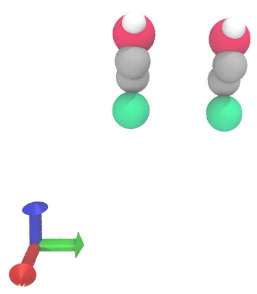
Evaluating CBMC Trials
Rotations of MeOH CCOH Dihedral in a 2 MeOH System

May 23, 2017

1 Evaluating Initial Energy

In this case we have two MeOH molecules separated by 5Å in the y direction.



Our MeOH molecules have the following coordinates:

$S\ (0.138, 0.0, -1.654)$	$S'\ (0.138, 5.0, -1.654)$
$C1\ (0.474, 0.0, 1.059)$	$C1'\ (0.474, 5.0, 1.059)$
$C2\ (-0.621, 0.0, -0.007)$	$C2'\ (-0.621, 5.0, -0.007)$
$O\ (-0.125, 0.0, 2.357)$	$O'\ (-0.125, 5.0, 2.357)$
$H\ (0.598, 0.0, 2.999)$	$H'\ (0.598, 5.0, 2.999)$

```
S=c(2,0.0,0.138,0.0,-1.654)
C1=c(1,0.0,0.474,0.0,1.059)
C2=c(1,0.265,-0.621,0.0,-0.007)
O=c(3,-0.700,-0.125,0.0,2.357)
H=c(4,0.435,0.598,0.0,2.999)
```

```r
S_2=c(2,0.0,0.138,5.0,-1.654)
C1_2=c(1,0.0,0.474,5.0,1.059)
C2_2=c(1,0.265,-0.621,5.0,-0.007)
O_2=c(3,-0.700,-0.125,5.0,2.357)
H_2=c(4,0.435,0.598,5.0,2.999)
```

Based on the number of atoms the total number of pairwise interactions are:

$$\binom{N_{atoms}}{2} = \binom{10}{2} = 45$$

Subtracting off the excluded intramolecular forces we have:

$$\binom{N_{atoms}}{2} - 2 \cdot \binom{N_{atoms}/2}{2} + 2 = \binom{10}{2} - 2 \cdot \binom{5}{2} + 2 = 27$$

The LJ Parameters are:

Elements	Epsilons (kcal/mol)	Sigmas (A)
C	0.118	3.905
S	0.397	4.250
O	0.200	2.850
H	0.000	1.780

Calculating the energies of all possible combinations we get:

```r
distance=function(atom1,atom2){
  return(norm(atom1[3:5]-atom2[3:5],type="2"))
}
energy=function(atom1,atom2){
  r=distance(atom1,atom2)
  sigma = (sigmas[atom1[1]]+sigmas[atom2[1]])/2
  epsilon = sqrt(epsilons[atom1[1]]*epsilons[atom2[1]])
  return(4*epsilon*((sigma/r)^12-(sigma/r)^6))
}
atoms = list(S,C1,C2,O,H,S_2,C1_2,C2_2,O_2,H_2)
atom_combos = combn(atoms,2)
distances=mapply(distance,atom_combos[1,],atom_combos[2,])
print(distances)

##  [1] 2.7337273 1.8134746 4.0196132 4.6756827 5.0000000 5.6985318 5.3187113
##  [8] 6.4153948 6.8455832 1.5281953 1.4295471 1.9439588 5.6985318 5.0000000
## [15] 5.2283249 5.2003466 5.3646040 2.4154735 3.2437628 5.3187113 5.2283249
## [22] 5.0000000 5.5528832 5.9600333 0.9668987 6.4153948 5.2003466 5.5528832
## [29] 5.0000000 5.0926312 6.8455832 5.3646040 5.9600333 5.0926312 5.0000000
## [36] 2.7337273 1.8134746 4.0196132 4.6756827 1.5281953 1.4295471 1.9439588
## [43] 2.4154735 3.2437628 0.9668987
```

```
  vdw_energies = mapply(energy,atom_combos[,distances>=5][1,],atom_combos[,distances>=5][2,
  print(vdw_energies)

##  [1] -0.37343756 -0.10065368 -0.14015464 -0.03144736  0.00000000
##  [6] -0.10065368 -0.08280619 -0.06771469 -0.04265897  0.00000000
## [11] -0.14015464 -0.06771469 -0.08280619 -0.02953994  0.00000000
## [16] -0.03144736 -0.04265897 -0.02953994 -0.02649616  0.00000000
## [21]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000

  sprintf("Total intermolecular Van der Waals Energy is: %4.4f kcal/mol",sum(vdw_energies))

## [1] "Total intermolecular Van der Waals Energy is: -1.3899 kcal/mol"
```

Therefore the total energy is given as the total intermolecular energy plus the intramolecular energy of the two MeOHs:

$$\begin{aligned}\text{Total Van der Waals Energy} &= \text{Intermolecular Energy} + \text{Intramolecular Energy}\\ &= -1.3898847 + -0.2812 = -1.6710847 \text{ kcal/mol}\end{aligned}$$

Now we use the following code to calculate the total coulombic energy:

```
  coul_energy=function(atom1,atom2){
    r=distance(atom1,atom2)
    k=0.2
    return(332.0636*((atom1[2]*atom2[2])/r)*exp(-k*r))
  }
  charged_atoms = list(C2,O,H,C2_2,O_2,H_2)
  charge_combos = combn(charged_atoms,2)
  distances=mapply(distance,charge_combos[1,],charge_combos[2,])
  print(distances)

##  [1] 2.4154735 3.2437628 5.0000000 5.5528832 5.9600333 0.9668987 5.5528832
##  [8] 5.0000000 5.0926312 5.9600333 5.0926312 5.0000000 2.4154735 3.2437628
## [15] 0.9668987

  coul_energies = mapply(coul_energy,charge_combos[,distances>=5][1,],charge_combos[,distanc
  print(sum(coul_energies))

## [1] 0.5628225
```

The coulombic energy calculated here is:

$$\boxed{\text{Total Coulombic Energy} = 0.5628225 \text{ kcal/mol}}$$

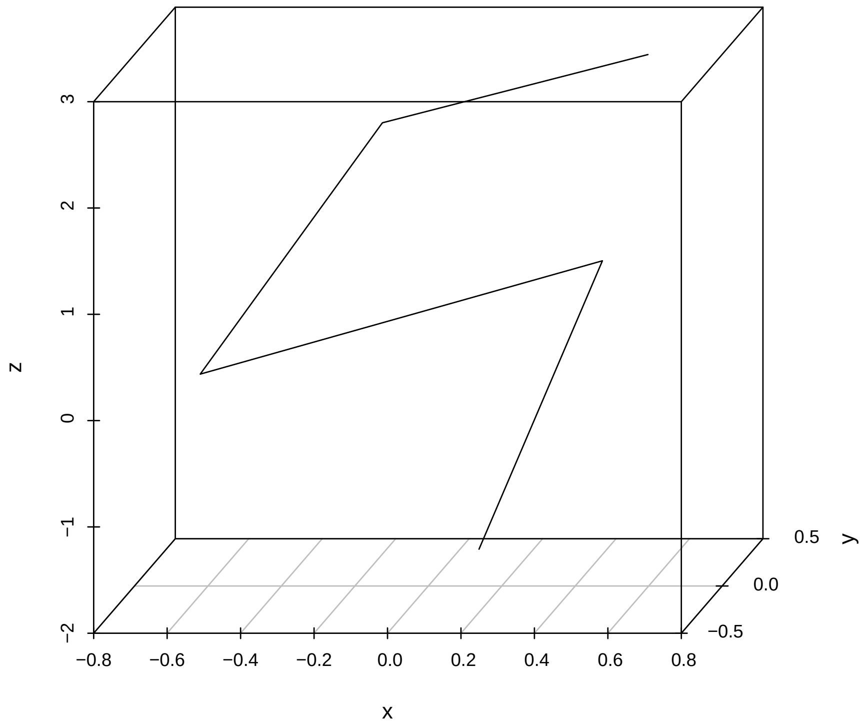
Then total energy is:

$$-1.1082622 \text{ kcal/mol}$$

2 Rotating CCOH Dihedral π Radians

The axis of rotation for the $CCOH$ dihedral is:

```
  library(scatterplot3d)
  axis = O[3:5]-C2[3:5]
  message(sprintf("The axis of rotation is: %s",paste(axis,collapse=", ")))

## The axis of rotation is:  0.496, 0, 2.364

  normaxis = axis/sqrt(sum(axis^2))
  message(sprintf("The normed axis of rotation is: %s",paste(normaxis,collapse = ", ")))

## The normed axis of rotation is:  0.205342766021818, 0, 0.978690118700761

  #Plot the molecule using scatterplot3d
  x = c(S[3],C1[3],C2[3],O[3],H[3])
  y = c(S[4],C1[4],C2[4],O[4],H[4])
  z = c(S[5],C1[5],C2[5],O[5],H[5])
  #data_frame = data.frame(x,y,z)
  scatterplot3d(x,y,z,type = "l")
```



The quaternion that defines a π radian rotation about this axis is:

$$\vec{q} = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2}) \cdot \frac{1}{\|a\|}\vec{a} = \cos(\frac{\pi}{2}) + \sin(\frac{\pi}{2}) \cdot \frac{1}{\|a\|}\vec{a}$$

This turns out to be:

```
library(onion)
theta = pi
q = cos(theta/2) + sin(theta/2) * normaxis[1] * Hi + sin(theta/2) * normaxis[2] *
    Hj + sin(theta/2) * normaxis[3] * Hk
message(sprintf("The quaternion of rotation is %s", paste(q, collapse = ", ")))

## The quaternion of rotation is 6.12303176911189e-17, 0.205342766021818,
0, 0.978690118700761
```

Finally we get the vectors we wish to rotate:

```
v1 = H[3:5] - O[3:5]
message(sprintf("Vector 1 is %s", paste(v1, collapse = ", ")))

## Vector 1 is 0.723, 0, 0.642
```

Applying the rotation quaternions we get:

```
v1_q = 0 + v1[1] * Hi + v1[2] * Hj + v1[3] * Hk
v1_rot_q = (q * v1_q) * Conj(q)
v1_rot = c(i(v1_rot_q), j(v1_rot_q), k(v1_rot_q))
message(sprintf("v1 rotated pi radians is now %s", paste(v1_rot, collapse = ", ")))

## v1 rotated pi radians is now -0.403986921956798, 7.05082905677622e-17,
0.878457492931714
```

This translates to final coordinates of O and H of:

```
Hnew = c(4, 0.435, v1_rot + O[3:5])
message(sprintf("Hnew is %s", paste(Hnew, collapse = ",")))

## Hnew is 4,0.435,-0.528986921956798,7.05082905677622e-17,3.23545749293171

# Plot the new dihedral
xnew = c(S[3], C1[3], C2[3], O[3], Hnew[3])
ynew = c(S[4], C1[4], C2[4], O[4], Hnew[4])
znew = c(S[5], C1[5], C2[5], O[5], Hnew[5])

xtotal = c(x, xnew)
ytotal = c(y, ynew)
ztotal = c(z, znew)
category = c(rep("blue", 5), rep("red", 5))
data_frame = data.frame(xtotal, ytotal, ztotal)
data_frame$fac = factor(rep(LETTERS[1:2], each = 5))
print(c(as.numeric(data_frame$xtotal)))

##  [1]  0.1380000  0.4740000 -0.6210000 -0.1250000  0.5980000  0.1380000
##  [7]  0.4740000 -0.6210000 -0.1250000 -0.5289869

scatterplot3d(x = data_frame$xtotal, y = data_frame$ytotal, z = data_frame$ztotal)

## Error in plot.window(c(x1, x2), c(z.min, z.max + yz.f * y.max)):
need finite 'xlim' values
```

```
# ,type = 'l',color=as.numeric(data_frame£fac)
```

The calculated dihedral angle from these new coordinates is:

```
library(pracma)

##
## Attaching package:  'pracma'
## The following object is masked from 'package:onion':
##
##     Norm

b1 = C2[3:5] - C1[3:5]
b2 = O[3:5] - C2[3:5]
b3 = Hnew[3:5] - O[3:5]
b2norm = b2/sqrt(sum(b2^2))
```

```
n1 = cross(b1, b2)/sqrt(sum(cross(b1, b2)^2))
n2 = cross(b3, b2)/sqrt(sum(cross(b2, b3)^2))
m1 = cross(n1, n2)
angle = atan2((m1 %*% b2norm), (n1 %*% n2))
message(sprintf("The calculated dihedral angle of the resulting rotation is: %4.2f degrees",
    angle * 180/pi))

## The calculated dihedral angle of the resulting rotation is:  -0.00
degrees
```

The new energies are:

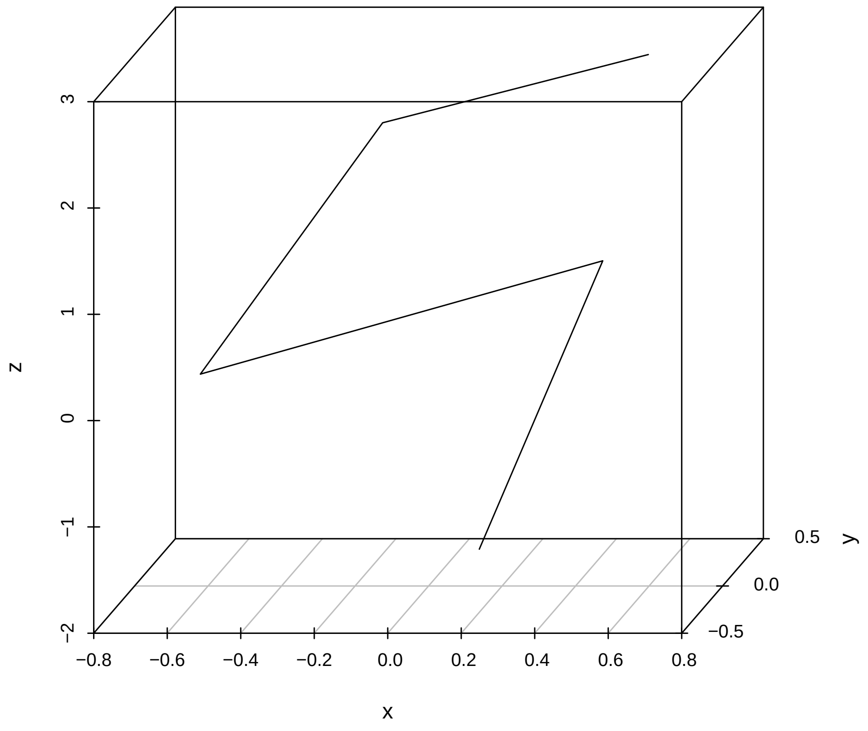
```
  atoms = list(S,C1,C2,O,Hnew,S_2,C1_2,C2_2,O_2,H_2)
  atom_combos = combn(atoms,2)
  distances=mapply(distance,atom_combos[1,],atom_combos[2,])
  print(distances)

##  [1] 2.7337273 1.8134746 4.0196132 4.9347407 5.0000000 5.6985318 5.3187113
##  [8] 6.4153948 6.8455832 1.5281953 1.4295471 2.3964453 5.6985318 5.0000000
## [15] 5.2283249 5.2003466 5.3646040 2.4154735 3.2437628 5.3187113 5.2283249
## [22] 5.0000000 5.5528832 5.9600333 0.9668987 6.4153948 5.2003466 5.5528832
## [29] 5.0000000 5.0926312 7.0250741 5.5446325 5.9600333 5.0926312 5.1308880
## [36] 2.7337273 1.8134746 4.0196132 4.6756827 1.5281953 1.4295471 1.9439588
## [43] 2.4154735 3.2437628 0.9668987

  vdw_energies = mapply(energy,atom_combos[,distances>=5][1,],atom_combos[,distances>=5][2,
  print(vdw_energies)

##  [1] -0.37343756 -0.10065368 -0.14015464 -0.03144736  0.00000000
##  [6] -0.10065368 -0.08280619 -0.06771469 -0.04265897  0.00000000
## [11] -0.14015464 -0.06771469 -0.08280619 -0.02953994  0.00000000
## [16] -0.03144736 -0.04265897 -0.02953994 -0.02649616  0.00000000
## [21]  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000

  print(sum(vdw_energies))+(2*-0.1406)

## [1] -1.389885
## [1] -1.671085

  charged_atoms = list(C2,O,Hnew,C2_2,O_2,H_2)
  charge_combos = combn(charged_atoms,2)
  distances=mapply(distance,charge_combos[1,],charge_combos[2,])
  print(distances)

##  [1] 2.4154735 3.2437628 5.0000000 5.5528832 5.9600333 0.9668987 5.5528832
##  [8] 5.0000000 5.0926312 5.9600333 5.0926312 5.1308880 2.4154735 3.2437628
## [15] 0.9668987
```

```
  coul_energies = mapply(coul_energy,charge_combos[,distances>=4.4][1,],charge_combos[,dista
  print(coul_energies)

## [1]  1.715728 -3.653670  1.949960 -3.653670 11.971618 -7.170113  1.949960
## [8] -7.170113  4.388782

  print(sum(coul_energies))

## [1] 0.3284828

  sprintf("Total energy is %4.8f kcal/mol",sum(coul_energies)+sum(vdw_energies)+(2*-0.1406)

## [1] "Total energy is -1.34260189 kcal/mol"
```

3 Rotating CCOH Dihedral $\frac{\pi}{2}$ Radians

The axis of rotation for the $CCOH$ dihedral is:

```
  library(scatterplot3d)
  axis = O[3:5]-C2[3:5]
  message(sprintf("The axis of rotation is: %s",paste(axis,collapse=", ")))

## The axis of rotation is:  0.496, 0, 2.364

  normaxis = axis/sqrt(sum(axis^2))
  message(sprintf("The normed axis of rotation is: %s",paste(normaxis,collapse = ", ")))

## The normed axis of rotation is:  0.205342766021818, 0, 0.978690118700761

  #Plot the molecule using scatterplot3d
  x = c(S[3],C1[3],C2[3],O[3],H[3])
  y = c(S[4],C1[4],C2[4],O[4],H[4])
  z = c(S[5],C1[5],C2[5],O[5],H[5])
  #data_frame = data.frame(x,y,z)
  scatterplot3d(x,y,z,type = "l")
```



The quaternion that defines a $\frac{\pi}{2}$ radian rotation about this axis is:

$$\vec{q} = \cos(\frac{\theta}{2}) + \sin(\frac{\theta}{2}) \cdot \frac{1}{\|a\|}\vec{a} = \cos(\frac{\pi}{4}) + \sin(\frac{\pi}{4}) \cdot \frac{1}{\|a\|}\vec{a}$$

This turns out to be:

```
library(onion)
theta = pi/2
q = cos(theta/2) + sin(theta/2) * normaxis[1] * Hi + sin(theta/2) * normaxis[2] *
    Hj + sin(theta/2) * normaxis[3] * Hk
message(sprintf("The quaternion of rotation is %s", paste(q, collapse = ", ")))

## The quaternion of rotation is 0.707106781186548, 0.14519926232163,
0, 0.692038419613575
```

Finally we get the vectors we wish to rotate:

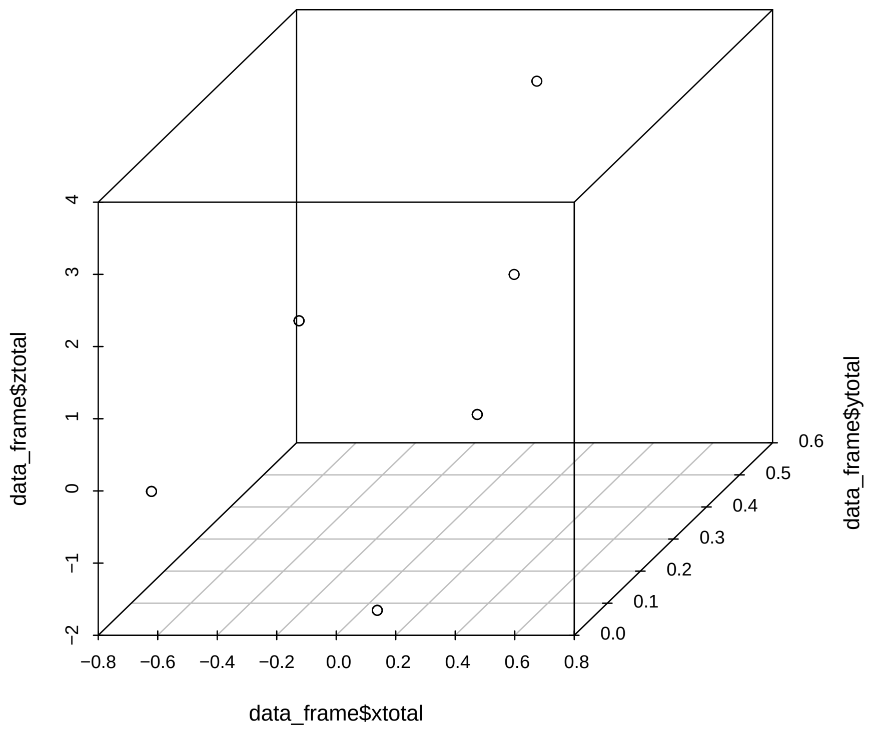
```
v1 = H[3:5] - O[3:5]
message(sprintf("Vector 1 is %s", paste(v1, collapse = ", ")))

## Vector 1 is 0.723, 0, 0.642
```

Applying the rotation quaternions we get:

```
v1_q = 0 + v1[1] * Hi + v1[2] * Hj + v1[3] * Hk
v1_rot_q = (q * v1_q) * Conj(q)
v1_rot = c(i(v1_rot_q), j(v1_rot_q), k(v1_rot_q))
message(sprintf("v1 rotated pi radians is now %s", paste(v1_rot, collapse = ", ")))

## v1 rotated pi radians is now 0.159506539021601, 0.575762900034643,
0.760228746465857
```

This translates to final coordinates of O and H of:

```
Hnew = c(4, 0.435, v1_rot + O[3:5])
message(sprintf("Hnew is %s", paste(Hnew, collapse = ",")))

## Hnew is 4,0.435,0.0345065390216012,0.575762900034643,3.11722874646586

# Plot the new dihedral
xnew = c(S[3], C1[3], C2[3], O[3], Hnew[3])
ynew = c(S[4], C1[4], C2[4], O[4], Hnew[4])
znew = c(S[5], C1[5], C2[5], O[5], Hnew[5])

xtotal = c(x, xnew)
ytotal = c(y, ynew)
ztotal = c(z, znew)
category = c(rep("blue", 5), rep("red", 5))
data_frame = data.frame(xtotal, ytotal, ztotal)
data_frame$fac = factor(rep(LETTERS[1:2], each = 5))
print(c(as.numeric(data_frame$xtotal)))

##  [1]  0.13800000  0.47400000 -0.62100000 -0.12500000  0.59800000
##  [6]  0.13800000  0.47400000 -0.62100000 -0.12500000  0.03450654

scatterplot3d(x = data_frame$xtotal, y = data_frame$ytotal, z = data_frame$ztotal)
```



```
# ,type = 'l',color=as.numeric(data_frame£fac)
```

The calculated dihedral angle from these new coordinates is:

```
library(pracma)
b1 = C2[3:5] - C1[3:5]
b2 = O[3:5] - C2[3:5]
b3 = Hnew[3:5] - O[3:5]
b2norm = b2/sqrt(sum(b2^2))
n1 = cross(b1, b2)/sqrt(sum(cross(b1, b2)^2))
n2 = cross(b3, b2)/sqrt(sum(cross(b2, b3)^2))
m1 = cross(n1, n2)
angle = atan2((m1 %*% b2norm), (n1 %*% n2))
message(sprintf("The calculated dihedral angle of the resulting rotation is: %4.2f degrees"
    angle * 180/pi))
```

```
## The calculated dihedral angle of the resulting rotation is:  -90.00
degrees
```

The new energies are:

```
  atoms = list(S,C1,C2,O,Hnew,S_2,C1_2,C2_2,O_2,H_2)
  atom_combos = combn(atoms,2)
  distances=mapply(distance,atom_combos[1,],atom_combos[2,])
  print(distances)

##  [1] 2.7337273 1.8134746 4.0196132 4.8069572 5.0000000 5.6985318 5.3187113
##  [8] 6.4153948 6.8455832 1.5281953 1.4295471 2.1819631 5.6985318 5.0000000
## [15] 5.2283249 5.2003466 5.3646040 2.4154735 3.2437628 5.3187113 5.2283249
## [22] 5.0000000 5.5528832 5.9600333 0.9668987 6.4153948 5.2003466 5.5528832
## [29] 5.0000000 5.0926312 6.5076270 4.8993197 5.4556730 4.4919110 4.4615442
## [36] 2.7337273 1.8134746 4.0196132 4.6756827 1.5281953 1.4295471 1.9439588
## [43] 2.4154735 3.2437628 0.9668987

  vdw_energies = mapply(energy,atom_combos[,distances>=5][1,],atom_combos[,distances>=5][2,
  print(vdw_energies)

##  [1] -0.37343756 -0.10065368 -0.14015464 -0.03144736  0.00000000
##  [6] -0.10065368 -0.08280619 -0.06771469 -0.04265897  0.00000000
## [11] -0.14015464 -0.06771469 -0.08280619 -0.02953994  0.00000000
## [16] -0.03144736 -0.04265897 -0.02953994 -0.02649616  0.00000000
## [21]  0.00000000  0.00000000

  print(sum(vdw_energies))+(2*-0.1406)

## [1] -1.389885
## [1] -1.671085

  charged_atoms = list(C2,O,Hnew,C2_2,O_2,H_2)
  charge_combos = combn(charged_atoms,2)
  distances=mapply(distance,charge_combos[1,],charge_combos[2,])
  print(distances)

##  [1] 2.4154735 3.2437628 5.0000000 5.5528832 5.9600333 0.9668987 5.5528832
##  [8] 5.0000000 5.0926312 5.4556730 4.4919110 4.4615442 2.4154735 3.2437628
## [15] 0.9668987

  coul_energies = mapply(coul_energy,charge_combos[,distances>=4.4][1,],charge_combos[,dista
  print(coul_energies)

## [1]  1.715728 -3.653670  1.949960 -3.653670 11.971618 -7.170113  2.356320
## [8] -9.166742  5.770186

  print(sum(coul_energies))
```

```
## [1] 0.1196177

  sprintf("Total energy is %4.8f kcal/mol",sum(coul_energies)+sum(vdw_energies)+(2*-0.1406))

## [1] "Total energy is -1.55146693 kcal/mol"
```