Calculating the Lennard-Jones Potential

General

The R tool that accomplishes this task is named LJpotential. LJpotential takes 4 required inputs (inputfile, x, y, z) and 2 optional inputs (epsilon_h, sigma). The x, y, and z inputs are the coordinates of the query point. The inputfile is the name of the PDB file from which we will calculate the Lennard-Jones potential. Because default PDB files include all atoms in an amino acid residue, not just the alpha carbons, we have written this code so that it can process either a default PDB file or a PDB file containing only alpha carbons. The code will run after loading the function from the LJpotential.rmd file. The calculated Lennard-Jones potential, expressed in kcal/mol, will display in the console.

Input Options for LJpotential:

Required: 1. inputfile - the name of the input file (1AHO.pdb) 2. x - x-coordinate of the query point, in angstroms 3. y - y-coordinate of the query point, in angstroms 4. z - z-coordinate of the query point, in angstroms

Optional: 5. epsilon_h - hydrophobic potential in kcal/mol 6. sigma - Van der Waals radius of amino acids in angstroms

If optional inputs are not specified, we will use a default epsilon_h value of 1 kcal/mol and a default sigma value of 5 angstroms, adapted from Veitshans, Klimov, & Thirumalai (1996).

Code

```
# Required inputs
inputfile <- "1AHO.pdb"
x <- 10
y <- 12
z <- 5

# Optional inputs
epsilon_h = 1  # in kcal/mol
sigma = 5 # in angstroms

LJpotential <- function(inputfile, x, y, z, epsilon_h, sigma){
  if(missing(epsilon_h)) {
    epsilon_h <- 1
    }
  if(missing(sigma)){
    sigma <- 5
  }

  # read in input file and clean for alpha carbons
  # NOTE: Code in this section adapted from Calvin's work in subproject 4.1
  no_col <- max(count.fields(inputfile))
  file <- read.table(inputfile, fill = TRUE, stringsAsFactors = FALSE, col.names = 1:no_col, skip = 1)
  atoms <- file[file$X1 == "ATOM",]
  alphaC <- atoms[atoms$X3 == "CA",]
  alphaC <- data.frame(alphaC[c(6, 4, 7:9)])
```

```
alphaC <- alphaC[!duplicated(alphaC[,1]),]
names(alphaC) <- c("ix", "res", "x", "y", "z")
alphaC$ix <- as.integer(alphaC$ix)
alphaC$x <- as.numeric(alphaC$x)
alphaC$y <- as.numeric(alphaC$y)
alphaC$z <- as.numeric(alphaC$z)
alphaC$res <- as.character(alphaC$res)

# classify alpha carbons with the BLN model
L = c("ARG", "LYS", "ASN", "GLU", "PRO", "ASP") # hydrophilic at pH 7
N = c("THR", "HIS", "GLY", "SER", "GLN") # neutral at pH 7
# assume that all others are hydrophobic (B)

for(i in 1:max(alphaC$ix)) {
  if(alphaC[i,2] %in% L == TRUE){
    alphaC$BLN[i] <- "L"
  }
  else if(alphaC[i,2] %in% N == TRUE){
    alphaC$BLN[i] <- "N"
  }
  else{
    alphaC$BLN[i] <- "B"
  }
}

# calculate distance from query point to alpha carbons
# NOTE: this section also adapted from Calvin's code
for(i in 1:max(alphaC$ix)) {
  vectdist <- data.frame(A = alphaC[i,3] - as.numeric(x), B = alphaC[i,4] - as.numeric(y),
                         C = alphaC[i,5] - as.numeric(y))
  normdist <- as.numeric(sqrt(vectdist[1]^2 + vectdist[2]^2 + vectdist[3]^2))
  alphaC$dist[i] <- normdist
}

# Calculate the Lennard-Jones potential

# for hydrophilic residues (lower epsilon)
for(i in 1:max(alphaC$ix)) {
  if(identical(alphaC[i,6],"L") == TRUE){
    epsilon <- 2/3.*epsilon_h
    VLJ <- 4*as.numeric(epsilon)*((as.numeric(sigma)/alphaC[i,7])^12 +
                                  (as.numeric(sigma)/alphaC[i,7])^6)
    alphaC$VLJ[i] <- VLJ
  }

  # for neutral residues (no repulsive term)
  else if(identical(alphaC[i,6],"N") == TRUE){
    epsilon <- epsilon_h
    VLJ <- 4*as.numeric(epsilon)*(as.numeric(sigma)/alphaC[i,7])^12
    alphaC$VLJ[i] <- VLJ
  }

  # for hydrophobic residues
```

```
    else if(identical(alphaC[i,6],"B") == TRUE){
      epsilon <- epsilon_h
      VLJ <- 4*as.numeric(epsilon)*((as.numeric(sigma)/alphaC[i,7])^12 +
                                      (as.numeric(sigma)/alphaC[i,7])^6)
      alphaC$VLJ[i] <- VLJ
    }
  }

  # Sum up all the LJ potentials on the query point and output
  LJ_tot <- sum(alphaC$VLJ)
  print(LJ_tot)
}
```