ProteinDynamics\_DimReduction

Arshan Munif (2262291)

2023-06-11

## About the R Markdown

This R Markdown file is designed to load and analyse protein trajectory data and conduct dimensionality reduction using PCA. To begin the user will read .pdb and .xtc files in this markdown

# Get working directory  
# The .pdb and .xtc files need to be in this directory. If not, R will not be able to locate and read the files and produce errors.  
getwd()

## [1] "/Users/arshanmunif/Documents/GitHub/MSRDimRed"

### Load Libraries

# Load necessary libraries  
library(bio3d)  
library(rioxdr)  
library(ggplot2)  
library(ggfortify)  
library(MASS)

### 1. Read trajectory and topology file, Select and extract C-alpha atoms, Reshape the data, Compute RMSD and PCA results

# Source the functions  
source('trajectory\_function.R')  
source('filter\_CAlpha.R')  
source('Reshape\_Data.R')  
source('extract\_coords.R')  
source('RMSD.R')  
source('PCA\_func.R')  
  
# Use the function to read trajectory(xtc) and topology (pdb) files  
trj\_obj <- read\_xtc("MD05\_ADK\_data/MD05\_ADK\_protein\_fit\_small.xtc", "MD05\_ADK\_data/MD05\_ADK\_protein.pdb")  
  
# Select c-alpha atoms  
inds <- select\_calpha(trj\_obj$pdb)  
# Extract c-alpha atom coordinates  
ca\_coords <- extract\_calpha\_coords(trj\_obj$trj, inds)  
  
  
# Reshape the data  
#reshaped\_coords <- reshape\_data(ca\_coords, inds)  
reshaped\_coords <- ca\_reshape(ca\_coords)  
  
  
# check the dimensions to ensure data prep and transformation were performed properly  
dim(ca\_coords)

## [1] 214 3 1000

dim(reshaped\_coords)

## [1] 1000 642

# Compute RMSD  
rmsd\_values <- compute\_rmsd(reshaped\_coords)

## Warning in bio3d::rmsd(coords): No indices provided, using the 214 non NA positions

mean(rmsd\_values)

## [1] 0.4013518

hist(rmsd\_values)

# Perform PCA using the PCA function  
pca\_result <- perform\_pca(reshaped\_coords)  
plot(pca\_result, col= 'orange', xlab = 'Principal Components', main = 'PCA Results Plot') # this plot will show the principal components in descending order of variance

### 2. Extract and plot principal components, compute explained variance, cumulative PVE, and visualise loadings

# Extract principal components scores  
scores <- pca\_result$x  
head(scores[,1:3]) # prints the first few lines for selected PCs

## PC1 PC2 PC3  
## [1,] 2.426249 -2.7448642 0.57288659  
## [2,] 2.380502 -0.1663568 0.58118578  
## [3,] 4.579609 -1.4895354 -0.25132701  
## [4,] 3.382745 0.3174489 0.67821057  
## [5,] 3.640786 -0.7834524 0.07873309  
## [6,] 3.859638 -0.4264242 0.02819847

tail(scores[,1:3]) # prints the last few lines for selected PCs

## PC1 PC2 PC3  
## [995,] -6.502444 -0.52635874 -0.7974073  
## [996,] -6.442920 0.14008907 -0.5538607  
## [997,] -6.217263 -0.03214594 -0.7473989  
## [998,] -6.135289 0.42734973 -0.8898183  
## [999,] -6.086973 0.03403481 -1.1859123  
## [1000,] -6.421491 0.04303629 -0.6130966

plot(scores, main = 'PC Scores') # plot the scores

# Generate color coded scores plot  
rgb.palette <- colorRampPalette(c("blue","green", "orange"), space = 'rgb')  
plot(scores[,1], scores[,2], pch = 20, col = rgb.palette(1000), xlab = 'PC1 Scores', ylab = 'PC2 Scores')

# Generate kernel density map  
a = kde2d(scores[,1], scores[,2], n = 1000)  
image(a)

# calculate proportion of variance explained (PVE) from std values  
pca\_result\_var <- pca\_result$sdev^2  
cat("Variance for selected PCs:", pca\_result\_var[1:3], "\n")

## Variance for selected PCs: 15.31831 1.374256 0.954099

pca\_result\_PVE <- pca\_result\_var / sum(pca\_result\_var)  
cat("Explained variance (PVE) for selected PCs:", pca\_result\_PVE[1:3], "\n")

## Explained variance (PVE) for selected PCs: 0.7295548 0.06545074 0.04544022

# calculate cumulative value of PVE for increasing number of additional principal components  
cumul\_var <- cumsum(pca\_result\_PVE)  
cat("Cumulative PVE for selected PCs:", cumul\_var[1:3], "\n")

## Cumulative PVE for selected PCs: 0.7295548 0.7950056 0.8404458

# get and inspect loadings principal components  
pca\_loadings <- pca\_result$rotation  
cat("Loadings for the selected PCs:", pca\_loadings[1:3], "\n")

## Loadings for the selected PCs: -0.03008377 0.01725033 -0.01398183

# plot the loadings for the PCs  
source('Loading\_Plot.R') # call loading plot function  
selected\_components <- c(1, 2, 3)  
plot\_loadings(pca\_loadings, selected\_components)

### 3. Generate PVE plot and Identify the top principal components representing signal

# Call the PVE plot function  
source('PVE\_Plot.R')  
  
# Generate plot with a 80% threshold line to inform feature extraction  
selected\_components <- 5 # insert the desired number of PCs that you want to be visualised in the scree plot  
select\_pc <- pve\_plot(cumul\_var, selected\_components)

## Number of components capturing 80 % variance: 3

# Print the number of components that explain 80% of variance  
cat('Selected number of components:', select\_pc)

## Selected number of components: 3

### 4. Genearte Bi-plot

library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

# call the biplot function  
source('biplot\_func.R')   
  
num\_scores <- 1000 # Change this to the desired number of scores that needs to be shown  
subset\_pca\_result <- pca\_result  
subset\_pca\_result$x <- pca\_result$x[1:num\_scores, ]  
  
# Call the pca\_biplot function  
pca\_biplot(subset\_pca\_result)

# visualise the variables  
fviz\_pca\_var(pca\_result, select.var = list(contrib = 100))

### 5. Project energy values on the PCA space

# read the energy table  
energy <- read.table("/Users/arshanmunif/Documents/GitHub/MSRDimRed/MD05\_small.energy")

# create a rgb colour coding function  
function (n)   
{  
 x <- ramp(seq.int(0, 1, length.out = n))  
 if (ncol(x) == 4L)   
 rgb(x[, 1L], x[, 2L], x[, 3L], x[, 4L], maxColorValue = 255)  
 else rgb(x[, 1L], x[, 2L], x[, 3L], maxColorValue = 255)  
}

# Generate a colour vector ordered by energy values  
col\_vec <- rgb.palette(1000)[order(energy$V1)]  
  
# Define a custom colour palette  
rgb.palette <- colorRampPalette(c("blue","green", "orange"),space = 'rgb')  
  
# Generate colour coded energy projection plot  
plot(scores, col = col\_vec, pch = 16, main = "Energy Projection on PCs")

### 6. Project RMSD Values on the PCA Space

# Define a custom colour palette  
rgb.palette <- colorRampPalette(c("blue", "green", "orange"), space = 'rgb')  
  
# Generate a colour vector ordered by RMSD values  
col\_vec\_2 <- rgb.palette(length(rmsd\_values))[order(rmsd\_values)]  
  
# project the rmsd on PCA plot  
plot(scores, col = col\_vec\_2, pch = 16, main = "RMSD Values Projected on PCs")

### 7. Project atomic distances on the PCA space

distance <-read.table("/Users/arshanmunif/Documents/GitHub/MSRDimRed/MD05\_ADK\_protein\_fit\_small.G55-P127.xvg", skip=17)  
names(distance) <- c("time", "distance")  
distance$col <- rgb(0,  
 (  
 (distance$distance - min(distance$distance)) /   
 (max(distance$distance) - min(distance$distance))),  
 0)  
  
# geneate histogram to understand distribution  
hist(distance$distance, 100)

hist(distance$distance, 1000)

# assign colours to distinguish open and closed states of protein  
distance$open <- 'red' # red depicts closed protein state  
distance[distance$distance > 2.1, 'open'] <- 'blue' # blue depicts open protein state  
  
# check colour assignment by viewing the first 6 rows in the data frame  
head(distance)

## time distance col open  
## 1 610 3.313 #00C000 blue  
## 2 1610 2.828 #009A00 blue  
## 3 2610 3.781 #00E500 blue  
## 4 3610 2.656 #008C00 blue  
## 5 4610 3.017 #00A900 blue  
## 6 5610 3.049 #00AB00 blue

# generate plot with the distance values projected on the PCA space  
plot(scores, col = distance$open, pch = 16, main = "Distances Projected on PCs")   
contour(a, add = T) # add contour overlay to the PCA plot to visualise the minima