Beginning Trajectory Analysis with Bio3D

Lars Skjaerven, Xin-Qiu Yao and Barry J. Grant
University of Michigan, Ann Arbor

November 8, 2013

1 Background

Bio3D[1] is an R package that provides interactive tools for structural bioinformatics. The primary focus of Bio3D is the analysis of bimolecular structure, sequence and simulation data.

Requirements: Detailed instructions for obtaining and installing the Bio3D package on various platforms can be found in the `Installing Bio3D` vignette available both on-line and from within the Bio3D package.

In addition to Bio3D the MUSCLE multiple sequence alignment program (available from http://www.drive5.com/muscle/) must be installed on your system and in the search path for executables. Please see the installation vignette for further details.

The aim of this document, termed a vignette[2] in R parlance, is to provide a brief task-oriented introduction to basic trajectory analysis with the Bio3D R package (Grant *et al.*, 2006). A number of other Bio3D package vignettes are available, including: `Installing Bio3D`, `Comparative protein structure analysis with Bio3D` and `Introduction to sequence conservation analysis with Bio3D` and `Enhanced methods for normal mode analysis with Bio3D`. In this vignette, `Beginning trajectory analysis with Bio3D`, we will demonstrate the use of several common Bio3D facilitates for the analysis of molecular dynamics trajectories.

1.1 Getting Started

Start R, load the Bio3D package and use the command `lbio3d()` to list the current functions available within the package:

```
library(bio3d)
lbio3d()
```

```
##   [1] "aa.index"          "aa123"             "aa2index"
##   [4] "aa2mass"           "aa321"             "aln2html"
##   [7] "angle.xyz"         "atom.index"        "atom.select"
##  [10] "atom2ele"          "atom2mass"         "atom2xyz"
##  [13] "binding.site"      "blast.pdb"         "bounds"
##  [16] "build.hessian"     "bwr.colors"        "chain.pdb"
##  [19] "cmap"              "com"               "com.xyz"
##  [22] "combine.sel"       "consensus"         "conserv"
##  [25] "convert.pdb"       "core.find"         "dccm"
##  [28] "dccm.enma"         "dccm.mean"         "dccm.nma"
##  [31] "dccm.xyz"          "deformation.nma"   "diag.ind"
##  [34] "difference.vector" "dist.xyz"          "dm"
##  [37] "dm.xyz"            "dssp"              "dssp.trj"
```

[1]The latest version of the package, full documentation and further vignettes (including detailed installation instructions) can be obtained from the main Bio3D website: http://thegrantlab.org/bio3d/

[2]This vignette contains executable examples, see `help(vignette)` for further details.

```
##  [40] "entropy"           "ff.anm"            "ff.calpha"
##  [43] "ff.calphax"        "ff.pfanm"          "ff.reach"
##  [46] "ff.sdenm"          "fit.xyz"           "fluct.nma"
##  [49] "formula2mass"      "gap.inspect"       "get.pdb"
##  [52] "get.seq"           "ide.filter"        "inner.prod"
##  [55] "is.gap"            "is.pdb"            "is.select"
##  [58] "kinesin"           "lbio3d"            "load.enmff"
##  [61] "mktrj.nma"         "mktrj.pca"         "mono.colors"
##  [64] "motif.find"        "nma"               "nma.pdbs"
##  [67] "normalize.vector"  "orient.pdb"        "overlap"
##  [70] "pairwise"          "pca.project"       "pca.tor"
##  [73] "pca.xyz"           "pca.xyz2z"         "pca.z2xyz"
##  [76] "pdb.annotate"      "pdb.summary"       "pdb2aln"
##  [79] "pdb2aln.ind"       "pdbaln"            "pdbfit"
##  [82] "pdbs2pdb"          "pdbseq"            "pdbsplit"
##  [85] "plot.bio3d"        "plot.blast"        "plot.core"
##  [88] "plot.dccm"         "plot.dccm2"        "plot.dmat"
##  [91] "plot.enma"         "plot.nma"          "plot.pca"
##  [94] "plot.pca.loadings" "plot.pca.score"    "plot.pca.scree"
##  [97] "plot.rmsip"        "print.core"        "print.enma"
## [100] "print.nma"         "print.pdb"         "print.rle2"
## [103] "print.sse"         "read.all"          "read.crd"
## [106] "read.dcd"          "read.fasta"        "read.fasta.pdb"
## [109] "read.mol2"         "read.ncdf"         "read.pdb"
## [112] "read.pdcBD"        "read.pqr"          "rgyr"
## [115] "rle2"              "rmsd"              "rmsd.filter"
## [118] "rmsf"              "rmsip"             "rot.lsq"
## [121] "sdENM"             "seq2aln"           "seqaln"
## [124] "seqaln.pair"       "seqbind"           "seqidentity"
## [127] "sse.bridges"       "store.atom"        "stride"
## [130] "struct.aln"        "summary.pdb"       "torsion.pdb"
## [133] "torsion.xyz"       "transducin"        "trim.pdb"
## [136] "unbound"           "vec2resno"         "view.dccm"
## [139] "view.modes"        "vmd.colors"        "wrap.tor"
## [142] "write.crd"         "write.fasta"       "write.ncdf"
## [145] "write.pdb"         "write.pqr"         "xyz2atom"
```

Detailed documentation and example code for each function can be accessed via the `help()` and `example()` commands (e.g. `help(read.pdb)`). You can also copy and paste any of the example code from the documentation of a particular function, or indeed this vignette, directly into your R session.

2 Reading Example Trajectory Data

A number of example data sets are shipped with the Bio3D package. The main purpose of including this data is to allow users to more quickly appreciate the capabilities of various Bio3D functions that would otherwise require potentially time consuming data generation. In the examples below we will input, process and analyze a molecular dynamics trajectory of Human Immunodeficiency Virus aspartic protease (HIVpr). This trajectory is stored in CHARMM/NAMD dcd format and has had all solvent and non C-alpha protein atoms excluded to reduce overall file size. Typically one works with all protein atoms, or at the very least all backbone atoms. Solvent however can often be excluded prior to Bio3D input - it just depends upon your particular analysis questions.

The code snippet below sets the file paths for the example HIVpr starting structure (pdbfile) and trajectory data (dcdfile).

```
dcdfile <- system.file("examples/hivp.dcd", package = "bio3d")
pdbfile <- system.file("examples/hivp.pdb", package = "bio3d")
```

Note that in the above example the `system.file()` command returns a character string corresponding to the file name of a PDB structure included with the Bio3D package. This is required as users may install the package in different locations. When using your own input files the `system.file()` command will not be required, for example

```
mydcdfile <- "/path/to/my/data/myfile.dcd"
```

```
dcd <- read.dcd(dcdfile)
pdb <- read.pdb(pdbfile)
```

The `read.dcd()` and `read.pdb()` commands processes the input files and returns their output to the new objects `dcd` and `pdb`. We can check the basic structure of these objects with the following commands:

```
print(pdb)
```

```
##
##  Call:  read.pdb(file = pdbfile)
##
##   Atom Count: 198
##
##    Total ATOMs#: 198
##      Protein ATOMs#: 198   ( Calpha ATOMs#: 198 )
##      Non-protein ATOMs#: 0   ( residues:  )
##      Chains#: 2   ( values: A B )
##
##    Total HETATOMs: 0
##      Residues HETATOMs#: 0   ( residues:  )
##      Chains#: 0   ( values:  )
##
##    Sequence:
##       PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQYD
##       QILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFPQITLWQRPLVTIKIGGQLKE
##       ALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQYDQILIEICGHKAIGTVLVGPTP
##       VNIIGRNLLTQIGCTLNF
##
## + attr: atom, het, helix, sheet, seqres,
##         xyz, xyz.models, calpha, call
```

```
length(pdb$xyz)
```

```
## [1] 594
```

```
dim(dcd)
```

```
## [1] 351 594
```

Note that the output of the `dim()` function is telling us that we have 351 trajectroy frames (or rows in our dcd matrix) and 594 coordinates (or x, y and z columns).

3 Trajectory Frame Superposition

In this simple example we select all C-alpha atoms for trajectory frame superposition.

```
ca.inds <- atom.select(pdb, elety = "CA")

##
## Build selection from input components
##
##       segid chain resno resid eleno elety
## Stest ""    ""    ""    ""    ""    "CA"
## Natom "198" "198" "198" "198" "198" "198"
##  *  Selected a total of: 198 intersecting atoms  *
```

The returned `ca.inds` object is a list containing atom and xyz numeric indices that we can now use to superpose all frames of the trajectory on the selected indices (in this case corresponding to all alpha Carbon atoms). For this we will with the `fit.xyz()` function.

```
xyz <- fit.xyz(fixed = pdb$xyz, mobile = dcd, fixed.inds = ca.inds$xyz, mobile.inds = ca.inds$xyz)
```

The above command performs the actual superposition and stores the new coordinates in the matrix object `xyz`. Note that the dimensions (i.e. number of rows and columns, which correspond to frames and coordinates respectively) of `xyz` match those of the input trajectory:

```
dim(xyz) == dim(dcd)

## [1] TRUE TRUE
```

4 Root Mean Square Deviation (RMSD)

RMSD is a standard measure of structural distance between coordinate sets and is implemented in the Bio3D function `rmsd()`.

```
rd <- rmsd(xyz[1, ca.inds$xyz], xyz[, ca.inds$xyz])
plot(rd, typ = "l", ylab = "RMSD", xlab = "Frame No.")
points(lowess(rd), typ = "l", col = "red", lty = 2, lwd = 2)
```

A quick histogram can be useful for examining the distribution of RMSD values.

```
hist(rd, breaks = 40, freq = FALSE, main = "RMSD Histogram", xlab = "RMSD")
lines(density(rd), col = "gray", lwd = 3)
```

```
summary(rd)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.85    2.02    2.18    2.22    4.47
```

5 Root Mean Squared Fluctuations (RMSF)

RMSF is an often used measure of conformational variance and is implemented in the Bio3D function `rmsf()`. This analysis will highlight the portions of structure that are fluctuating from their mean structure the most (and least).

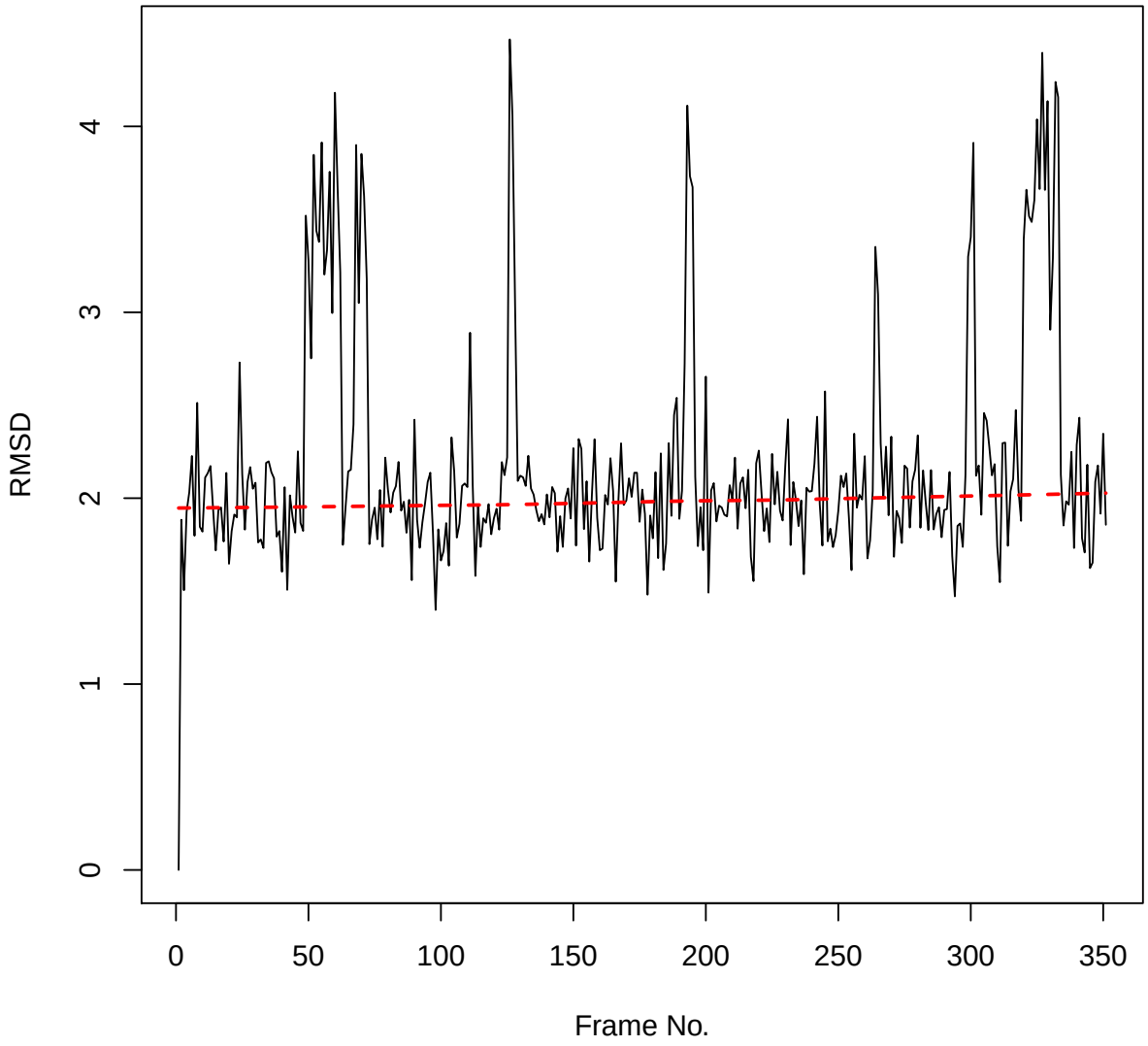


Figure 1: Simple time series of RMSD from the initial structure (note periodic jumps that we will later see correspond to transient openings of the flap regions of HIVpr)

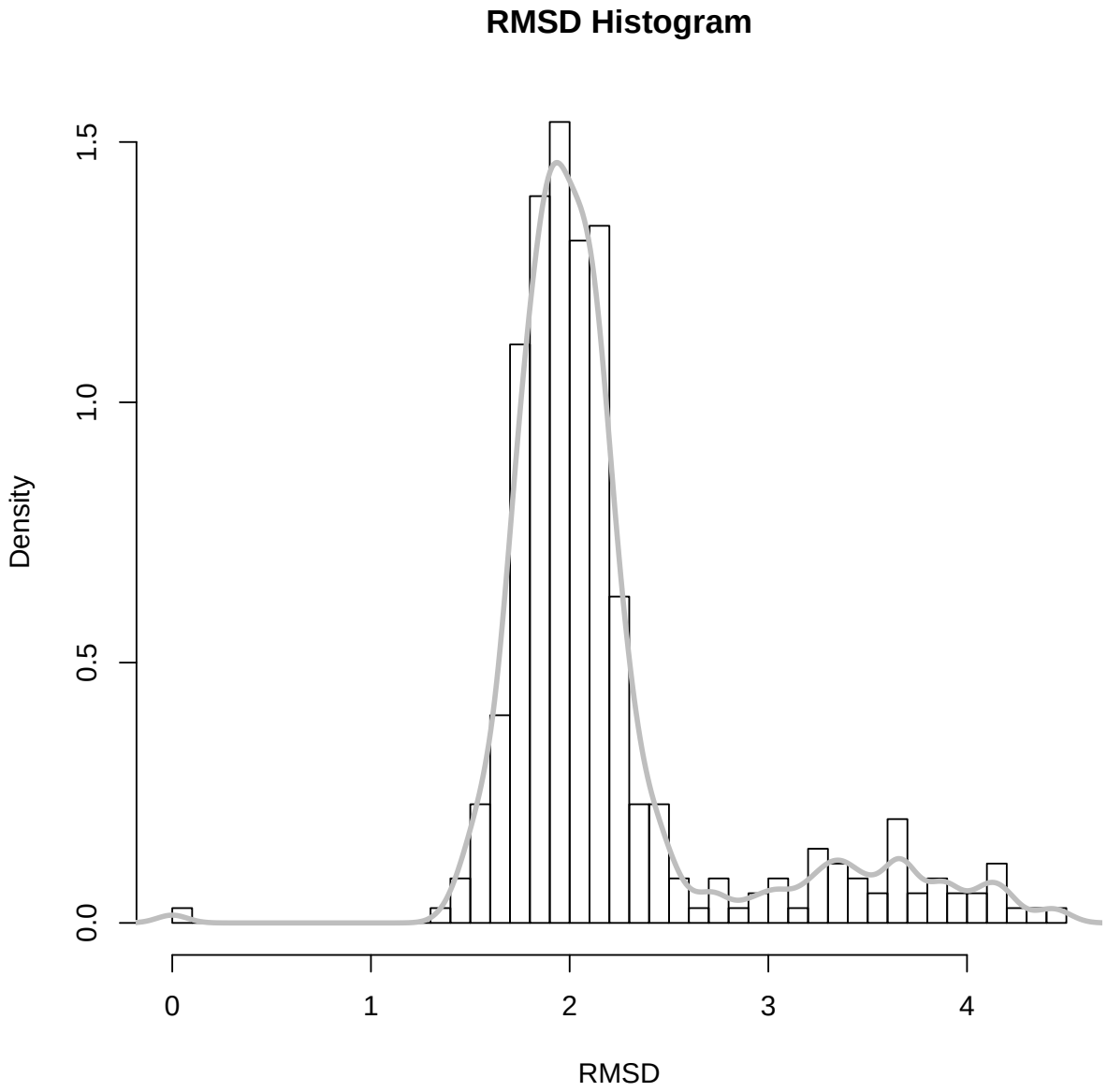


Figure 2: Note the spread of RMSD values and that the majority of sampled conformations are around 2 Angstroms from the starting structure

```
rf <- rmsf(xyz[, ca.inds$xyz])
plot(rf, ylab = "RMSF", xlab = "Residue Position", typ = "l")
```

6 Principal Component Analysis

PCA can be employed to examine the relationship between different conformations sampled during the trajectory and is implemented in the Bio3D functions `pca.xyz()` and `pca.tor()`. The application of PCA to both distributions of experimental structures and molecular dynamics trajectories will be covered in detail in other vignettes. Briefly, we will note here that this method can provide considerable insight into the nature of conformational differences with the resulting principal components (orthogonal eigenvectors) describing the axes of maximal variance of the distribution of structures. Projection of the distribution onto the subspace defined by the largest principal components results in a lower dimensional representation of the structural dataset. The percentage of the total mean square displacement (or variance) of atom positional fluctuations captured in each dimension is characterized by their corresponding eigenvalue (see the next figure). Experience suggests that 3–5 dimensions are often sufficient to capture over 70 percent of the total variance in a given family of experimental structures or indeed a standard molecular dynamics trajectory. Thus, a handful of principal components are sufficient to provide a useful description while still retaining most of the variance in the original distribution Grant *et al.* (2006).

A quick overview of the results of `pca.xyz()` can be obtained by calling `plot.pca()`

```
pc <- pca.xyz(xyz[, ca.inds$xyz])
plot(pc, col = bwr.colors(nrow(xyz)))
```

Note that there are distinct groupings of conformations along the PC1 plane (one centered around -30 and a second, larger grouping, at +5). The continuous color scale (from blue to whit to red) indicates that there are periodic jumps between these conformers throughout the trajectory. Below we perform a quick clustering in PC-space to further highlight these distinct conformers.

```
hc <- hclust(dist(pc$z[, 1:2]))
grps <- cutree(hc, k = 2)
plot(pc, col = grps)
```

Bellow we call `plot.bio3d()` to examine the contribution of each residue to the first two principal components.

```
plot.bio3d(pc$au[, 1], ylab = "PC1 (A)", xlab = "Residue Position", typ = "l")
points(pc$au[, 2], typ = "l", col = "blue")
```

To further aid interpretation, a PDB format trajectory can be produced that interpolates between the most dissimilar structures in the distribution along a given principal component. This involves dividing the difference between the conformers into a number of evenly spaced steps along the principal components, forming the frames of the output multi-model PDB trajectory. Such trajectories can be directly visualized in a molecular graphics program, such as VMD (Humphrey *et al.*, 1996). Furthermore, the interpolated structures can be analyzed for possible domain and shear movements with other Bio3D functions, or used as initial seed structures for reaction path refinement methods (note you will likely want to perform all heavy atom PCA for such applications).

```
p1 <- mktrj.pca(pc, pc = 1, b = pc$au[, 1], file = "pc1.pdb")
p2 <- mktrj.pca(pc, pc = 2, b = pc$au[, 2], file = "pc2.pdb")
```

You can also write these trajectory's as AMBER NetCDF format files with the `write.ncdf` function. To view the PDB trajectories in VMD just open the files in the normal way and display as tube representation for example (see figure below).

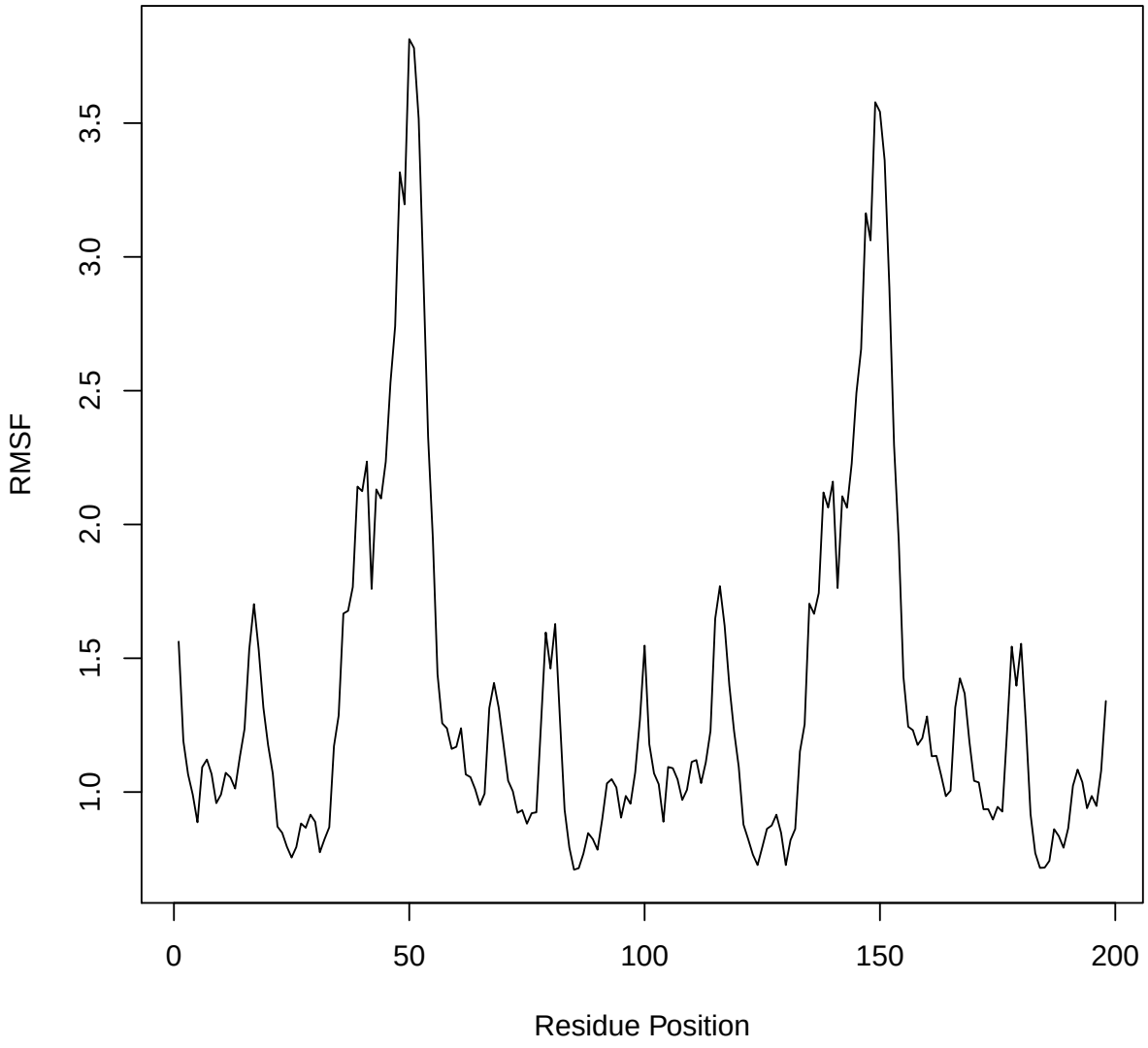


Figure 3: Residue-wise RMSF indicates regions of high mobility

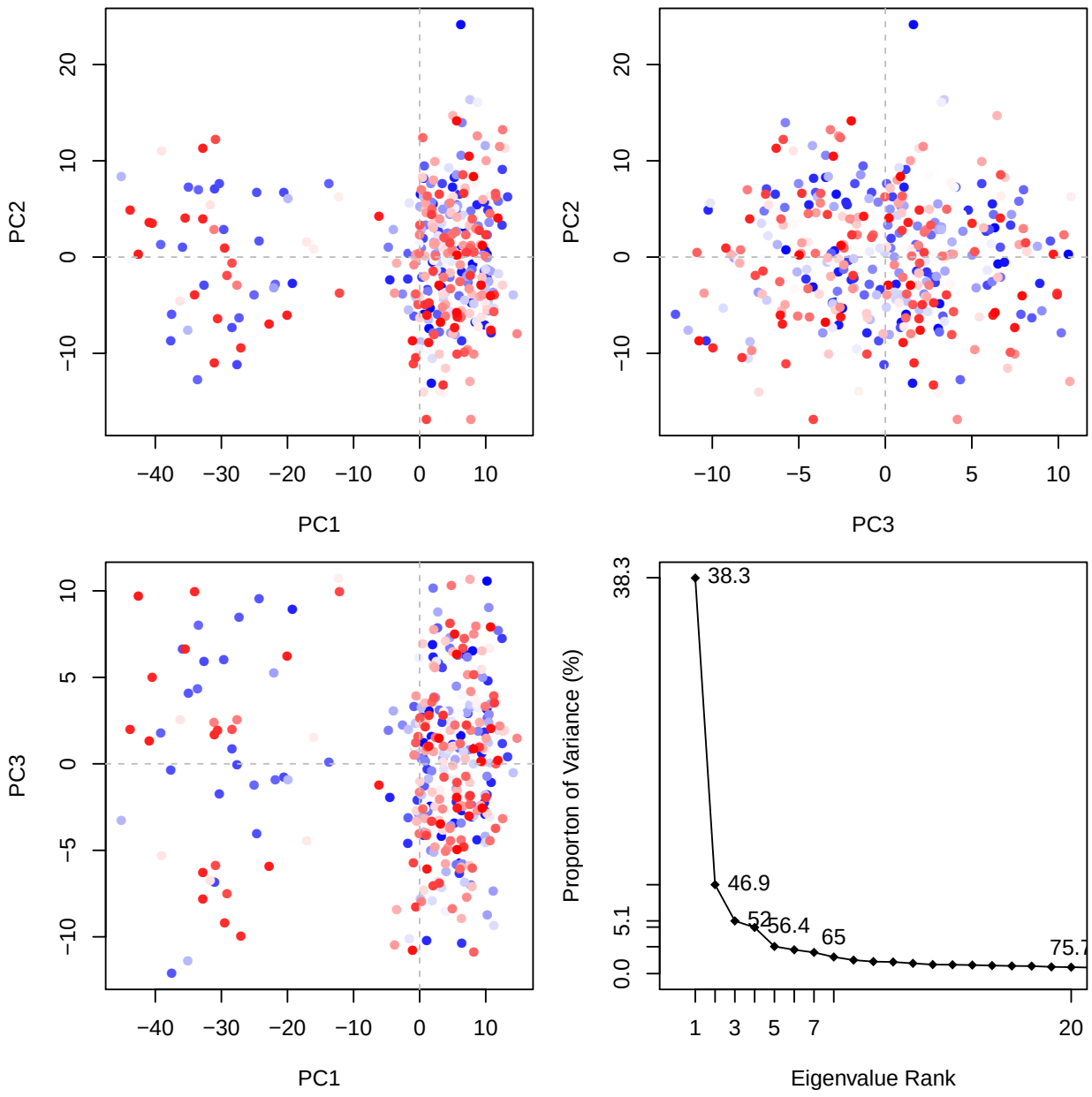


Figure 4: PCA results for our HIVpr trajectory with instantaneous conformations (i.e. trajectory frames) colored from blue to red in order of time

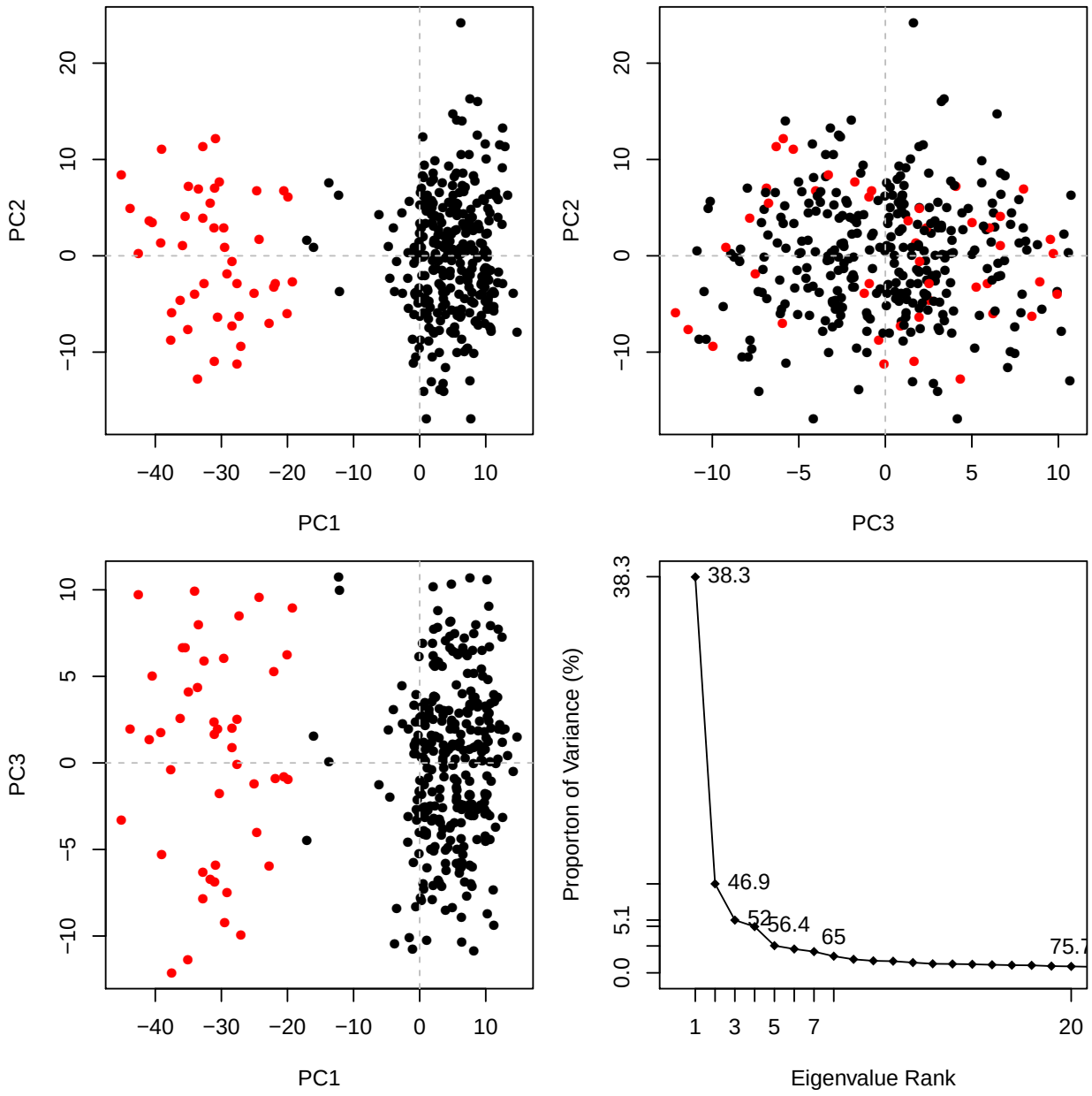


Figure 5: Simple clustering in PC subspace

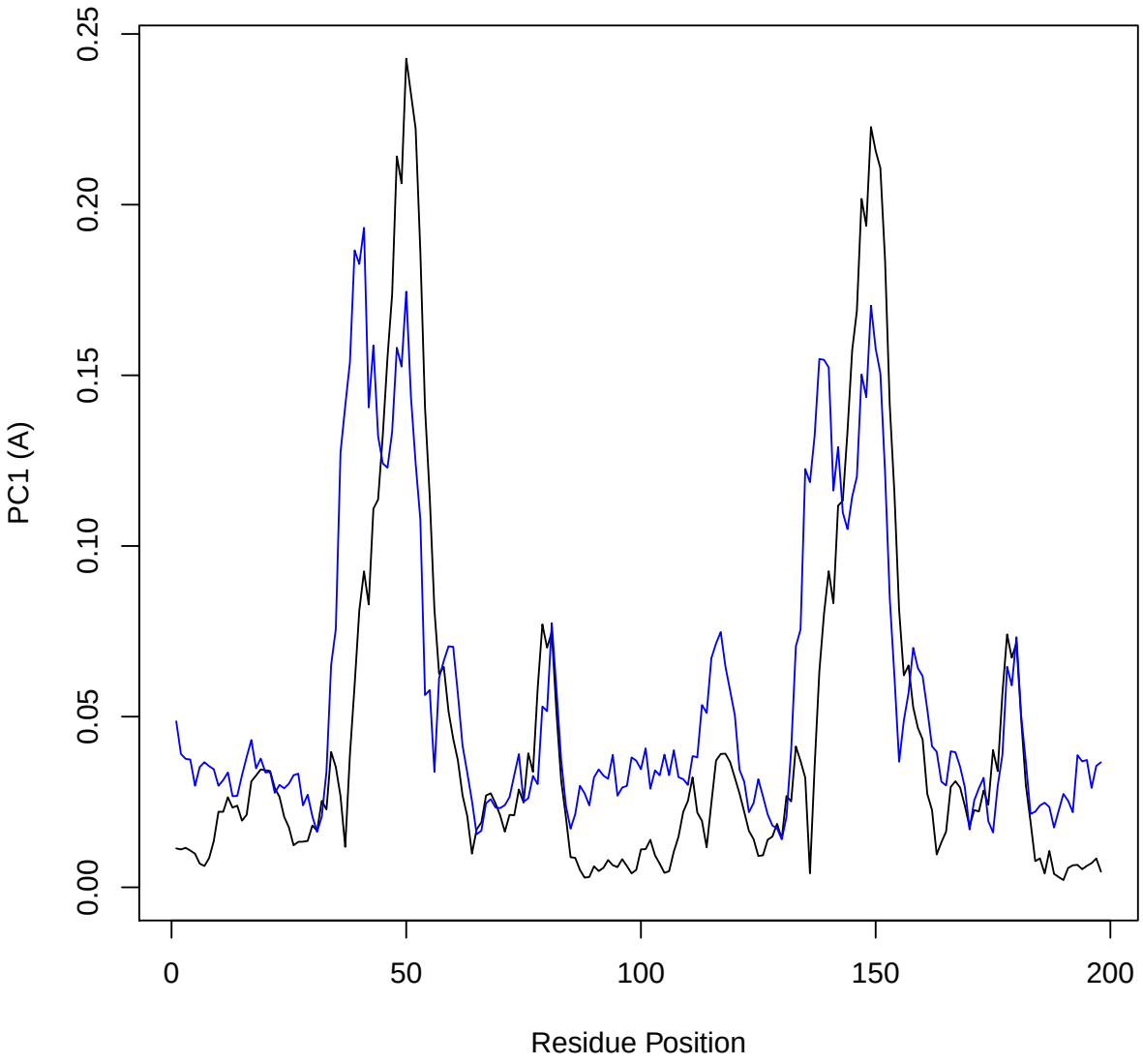
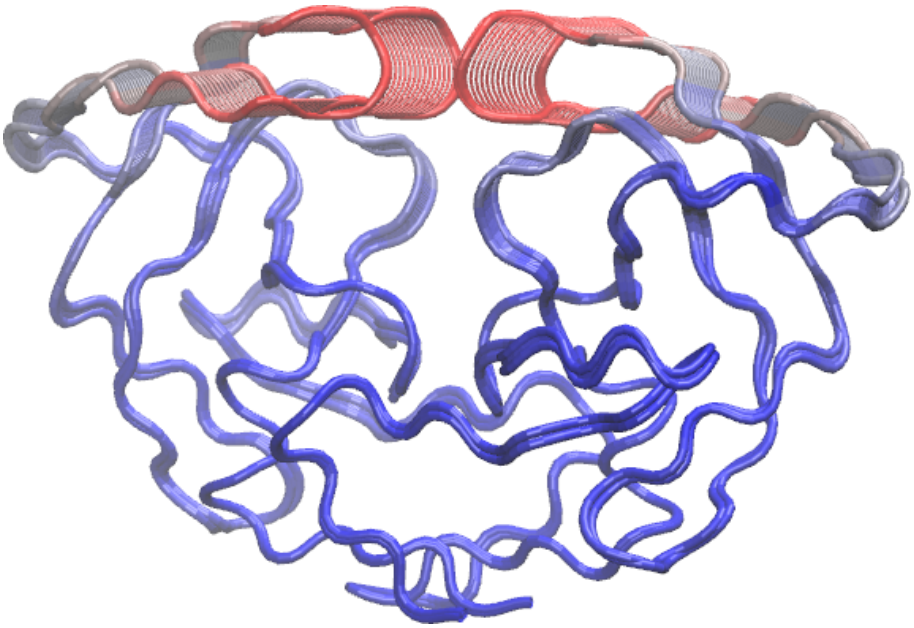


Figure 6: Residue-wise loadings for PC1 (black) and PC2 (blue)

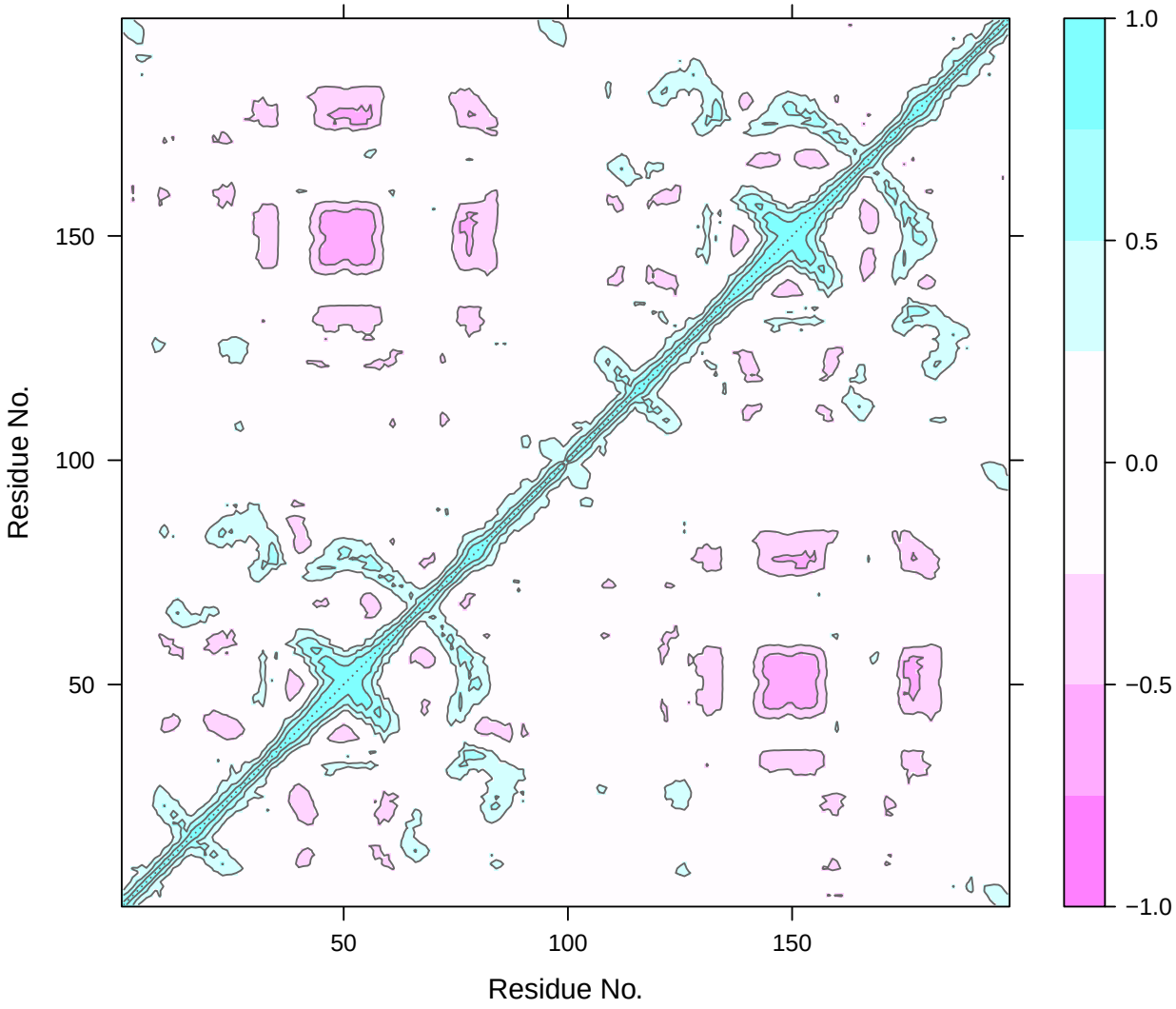
```
write.ncdf(p1, "trj_pc1.nc")
```



7 Cross-Correlation Analysis

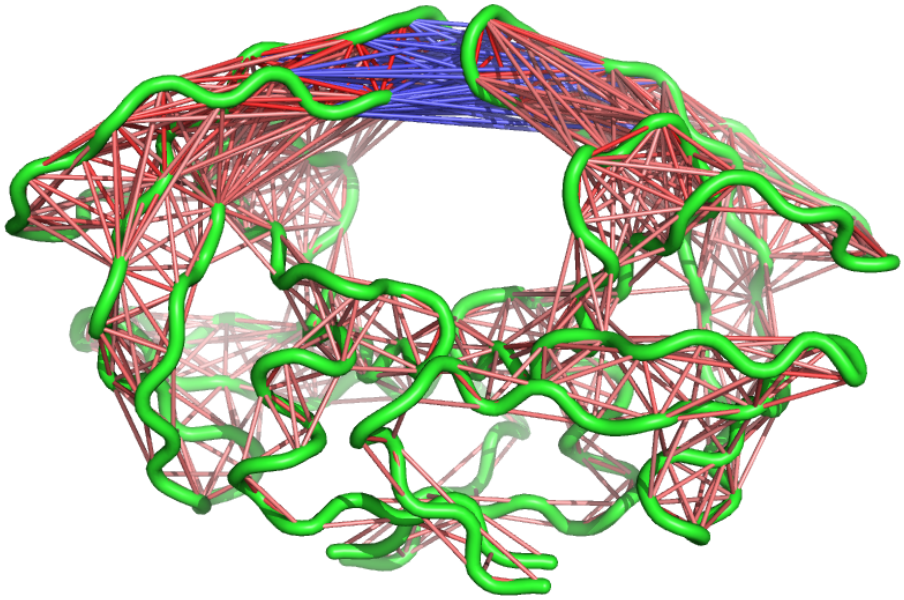
The extent to which the atomic fluctuations/displacements of a system are correlated with one another can be assessed by examining the magnitude of all pairwise cross-correlation coefficients. The Bio3D `dccm()` function returns a matrix of all atom-wise cross-correlations whose elements may be displayed in a graphical representation frequently termed a dynamical cross-correlation map, or DCCM.

```
cij <- dccm(xyz[, ca.inds$xyz])
plot(cij)
```



A 3D visualization of these correlations can be provided through the function `view.dccm()`

```
# View the correlations in pymol
view.dccm(cij, pdb, launch = TRUE)
```



See also the `Enhanced Methods for Normal Mode Analysis` for additional visualization examples. Also you might want to checkout the `Comparative Analysis of Protein Structures` vignette for relating results like these to available experimental data. The logical expansion of this analysis is described in the `Correlation Network Analysis` vignette.

8 Where to Next

If you have read this far, congratulations! We are ready to have some fun and move to other package vignettes that describe more interesting analysis including Correlation Network Analysis (where we will build and dissect dynamic networks form different correlated motion data), enhanced methods for Normal Mode Analysis (where we will explore the dynamics of large protein families and superfamilies), and advanced Comparative Structure Analysis (where we will mine available experimental data and supplement it with simulation results to map the conformational dynamics and coupled motions of proteins).

9 Document Details

This document is shipped with the Bio3D package in both Rnw and PDF formats. All code can be extracted and automatically executed to generate Figures and/or the PDF with the following commands:

```
library(knitr)
knit("Bio3D_md.Rnw")
tools::texi2pdf("Bio3D_md.tex")
```

Information About the Current Bio3D Session

```
sessionInfo()

## R version 3.0.2 (2013-09-25)
## Platform: x86_64-apple-darwin10.8.0 (64-bit)
##
## locale:
```

```
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  utils     datasets  grDevices methods
## [8] base
##
## other attached packages:
## [1] lattice_0.20-24 bio3d_2.0       knitr_1.5
##
## loaded via a namespace (and not attached):
## [1] codetools_0.2-8 digest_0.6.3    evaluate_0.5.1  formatR_0.9
## [5] highr_0.2.1     stringr_0.6.2   tcltk_3.0.2     tools_3.0.2
```

References

Grant, B.J. and Rodrigues, A.P.D.C and Elsawy, K.M. and Mccammon, A.J. and Caves, L.S.D. (2006) **Bio3d: an R package for the comparative analysis of protein structures.** *Bioinformatics*, **22**, 2695–2696.

Humphrey, W., et al. (1996) **VMD: visual molecular dynamics.** *J. Mol. Graph*, **14**, 33–38