Comparative Sequence and Structure Analysis with Bio3D

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

November 12, 2013

1 Background

Bio3D[1] is an R package that provides interactive tools for the analysis of bimolecular structure, sequence and simulation data. The aim of this document, termed a vignette[2] in R parlance, is to provide a brief task-oriented introduction to comparative sequence and structure analysis with the Bio3D R package (**?**). Various Bio3D utilities are introduced to demonstrate how to read and write sequence and structure data, perform homologous protein search, X-ray structures annotation, atom selection, re-orientation, superposition, rigid core identification, clustering, torsion analysis, distance matrix analysis, structure and sequence conservation analysis, and principal component analysis. In addition, the document illustrates utility functions that enable the statistical and graphical power of the R environment to work with biological sequence and structural data.

Requirements: Detailed instructions for obtaining and installing the Bio3D package on various platforms can be found in the Installing Bio3D vignette available both online and from within the Bio3D package. To see available vignettes use the command:

```r
vignette(package = "bio3d")
```

Note that to follow along with this vignette the MUSCLE multiple sequence alignment program must be installed on your system and in the search path for executables. In addition, the DSSP secondary structure assignment program must be installed and in the search path. Please see the installation vignette for full details.

2 Getting Started

Start R, load the Bio3D package and use the command `demo("pdb")` and `demo("pca")` to get a quick feel for some of the tasks that we will be introducing in the following sections.

```r
library(bio3d)
demo("pdb")
demo("pca")
```

Side-note: Note that you will be prompted to hit the `RETURN` key at each step of the demo as this will allow you to see the particular functions being called. Also note that 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 to see how things work. You can also find this documentation online.

[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.

2.1 Read A PDB Structure

The code snippet below reads structure data online and returns an object `pdb` for further manipulation. The function `read.pdb()` is supplied with one input argument, the four letter RCSB Protein Data Bank (PDB) identifier `1tag`.

```
pdb <- read.pdb("1tag")

##   Note: Accessing online PDB file
##   HEADER    GTP-BINDING PROTEIN                     23-NOV-94   1TAG
```

Side-note: If you have PDB file stored on your local hard disk, use the path to that file as the argument of `read.pdb()`. For example:

```
pdb <- read.pdb("/path/to/my/data/myfile.pdb")
```

To examine the contents of the `pdb` object we can use the `attributes` command:

```
attributes(pdb)

## $names
## [1] "atom"       "het"        "helix"      "sheet"      "seqres"
## [6] "xyz"        "xyz.models" "calpha"     "call"
##
## $class
## [1] "pdb"
```

Most bio3d functions, including `read.pdb()`, return list objects whose components can be accessed in the standard way (see `help(read.pdb)` for further details). For example, to print coordinate data for the first three atoms in our newly created `pdb` object type:

```
pdb$atom[1:3, c("resno", "resid", "elety", "x", "y", "z")]

##      resno resid elety x        y        z
## [1,] "27"  "ALA" "N"   "38.238" "18.018" "61.225"
## [2,] "27"  "ALA" "CA"  "38.552" "16.715" "60.576"
## [3,] "27"  "ALA" "C"   "40.042" "16.687" "60.253"
```

Side-note: Simply type the name of the object `pdb` will call the function `print(pdb)`, which returns a summary of the structural data:

```
pdb

##
##  Call:  read.pdb(file = "1tag")
##
##   Atom Count: 2890
##
##    Total ATOMs#: 2521
##      Protein ATOMs#: 2521   ( Calpha ATOMs#: 314 )
##      Non-protein ATOMs#: 0   ( residues:  )
##      Chains#: 1   ( values: A )
##
```

```
##    Total HETATOMs: 369
##      Residues HETATOMs#: 342   ( residues: MG GDP HOH )
##      Chains#: 1   ( values: A )
##
##    Sequence:
##       ARTVKLLLLGAGESGKSTIVKQMKIIHQDGYSLEECLEFIAIIYGNTLQSILAIVRAMTT
##       LNIQYGDSARQDDARKLMHMADTIEEGTMPKEMSDIIQRLWKDSGIQACFDRASEYQLND
##       SAGYYLSDLERLVTPGYVPTEQDVLRSRVKTTGIIETQFSFKDLNFRMFDVGGQRSERKK
##       WIHCFEGVTCIIFIAALSAYDMVLVEDDEVNRMHESLHLFNSICN...<cut>...DIII
##
## + attr: atom, het, helix, sheet, seqres,
##         xyz, xyz.models, calpha, call
```

In the previous example we use numeric indices to access atoms 1 to 3. In a similar fashion the `atom.select()` function returns numeric indices that can be used for convenient data access:

```
inds <- atom.select(pdb, "calpha")

##
## Build selection from input string
##
##
##  Using selection 'string' keyword shortcut: calpha = //////CA/
##
##       segid  chain  resno  resid  eleno  elety
## Stest ""     ""     ""     ""     ""     "CA"
## Natom "2521" "2521" "2521" "2521" "2521" "314"
##  *  Selected a total of: 314 intersecting atoms  *

plot.bio3d(pdb$atom[inds$atom, "resno"], pdb$atom[inds$atom, "b"], sse = pdb,
    ylab = "Bfactor", xlab = "Residue Number")
```

The returned `inds` object is a list containing atom and xyz numeric indices (in this case corresponding to all alpha Carbon atoms). In the above example we use these indices to plot residue number vs B-factor along with a basic secondary structure schematic (Figure 1). As a further example of data access lets extract the sequence for the loop region (P-loop) between strand 3 (beta 1) and helix 1 in our `pdb` object.

```
loop <- pdb$sheet$end[3]:pdb$helix$start[1]
loop.inds <- atom.select(pdb, paste("///", paste(loop, collapse = ","), "///CA/",
    sep = ""))

##
## Build selection from input string
##
##       segid  chain  resno                       resid  eleno  elety
## Stest ""     ""     "35,36,37,38,39,40,41,42" ""     ""     "CA"
## Natom "2521" "2521" "49"                        "2521" "2521" "314"
##  *  Selected a total of: 8 intersecting atoms  *

pdb$atom[loop.inds$atom, "resid"]

## [1] "LEU" "GLY" "ALA" "GLY" "GLU" "SER" "GLY" "LYS"

aa321(pdb$atom[loop.inds$atom, "resid"])

## [1] "L" "G" "A" "G" "E" "S" "G" "K"
```

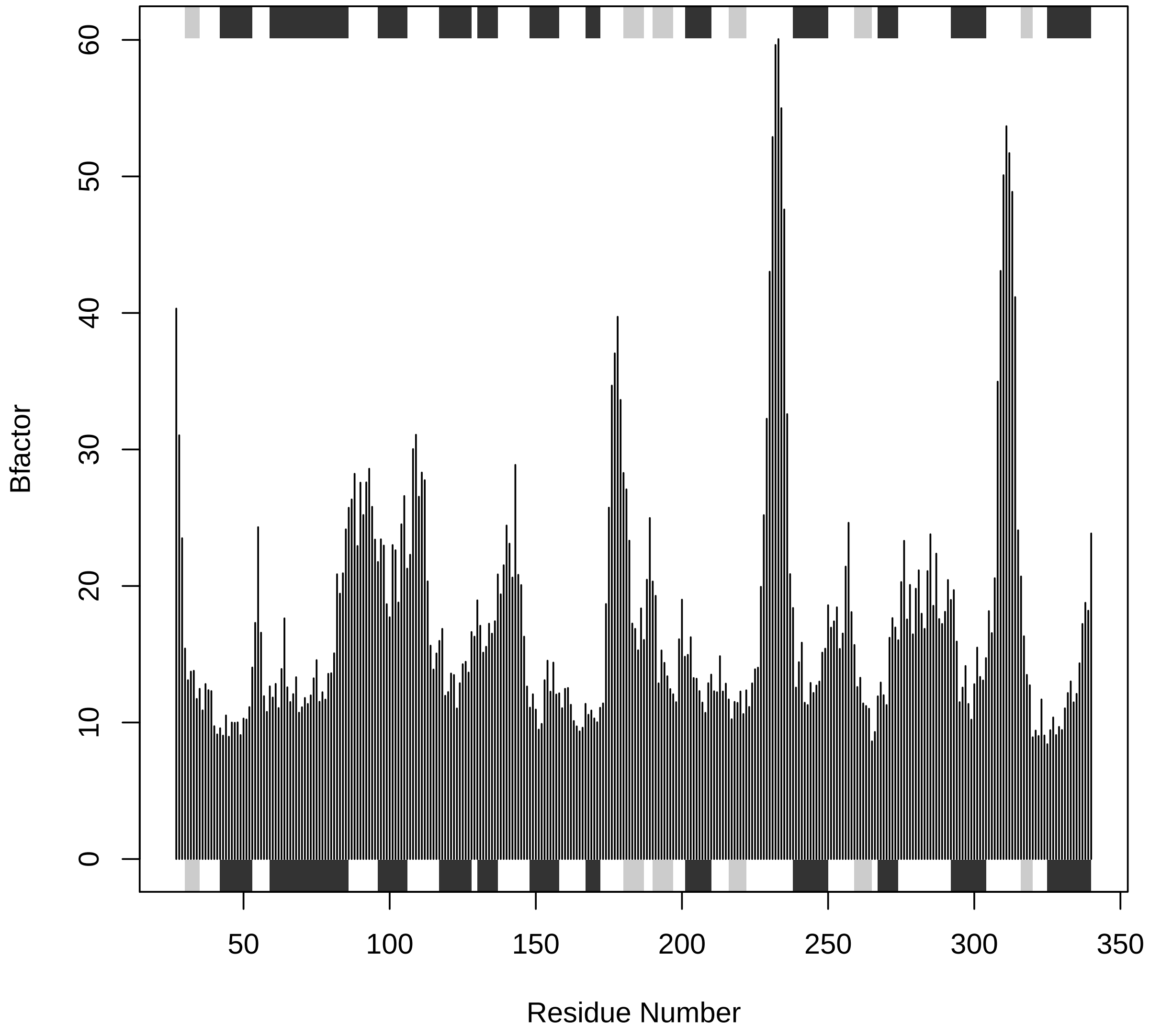


Figure 1: B-factor along with residue numbers

In the above example the residue numbers in the `sheet` and `helix` components of `pdb` are accessed and used in a subsequent atom selection. The `aa321()` function converts between three-letter and one-letter IUPAC amino acid codes.

Question: How would you do to select all backbone atoms? HINT: help(atom.select) and check out the keyword `string`.

Question: How would you do to select all sidechain atoms? HINT: Consider two steps and check out `help(combine.sel)`.

2.2 Read Example Data

A number of example datasets are included with the bio3d package. The main purpose of including this data (which may be generated by the user by following the extended examples documented within the various bio3d functions) is to allow users to more quickly appreciate the capabilities of functions that would otherwise require data input and processing before execution.

The Transducin Heterotrimeric G Proteins For the worked examples in the current document we will utilize the included transducin dataset. A related dataset formed the basis of the work described in (**?**). Briefly, heterotrimeric G proteins are molecular switches that turn on intracellular signaling cascades in response to the activation of G protien coupled receptors (GPCRs). Receptor activation by extracellular stimuli promotes a cycle of GTP binding and hydrolysis on the G protein alpha subunit.

Heterotrimeric G proteins undergo cycles of GTP-dependent conformational rearrangements and alterations of their oligomeric abg form to convey receptor signals to downstream effectors. Interaction with activated receptor promotes the exchange of GDP for GTP on the G protein alpha subunit (Ga) and its separation from its beta-gammar subunit partners (Gbg). The current dataset consists of transducin (including Gt and Gi/o) alpha subunit sequence and structural data and can be loaded with the command `data(transducin)`:

```
data(transducin)
attach(transducin)
```

Side-note: The transducin dataset can be assembled from scratch with the `get.pdb()`, `pdbsplit`, and `pdbaln()` commands bellow, which only show as an example the procedure for a subset of transducin PDB ids (See Section 3 for more details):

```
## Download and split transducin PDB files
ids <- c("1TND_B", "1AGR_A", "1FQJ_A", "1TAG_A", "1GG2_A", "1KJY_A")
raw.files <- get.pdb(ids)
files <- pdbsplit(raw.files, ids)
## Alignment
pdbs <- pdbaln(files)
```

2.2.1 The Kinesin Molecular Motor

We also utilize the included kinesin dataset especially for demostrating sequence analysis for a protein family with diverse sequences. A related dataset formed the basis of the work described in (**?**). Briefly, kinesins are molecular motor proteins responsible for the ATP dependent transport of cellular cargo along microtubules. Kinesin family members have been found in all eukaryotic organisms, where they contribute to the transport of molecules and organelles, organisation and maintenance of the cytoskeleton, and the segregation of genetic material during mitosis and meiosis.

The defining attribute of kinesin family members is the possession of one or more globular motor domains. These ∼350 residue domains are responsible for ATP hydrolysis, microtubule binding and force production. The current dataset consists of kinesin motor domain sequence and structural data and can be loaded with the command `data(kinesin)`:

```
data(kinesin)
attach(kinesin)
```

Side-note: To check out the components, and their meanings, in the transducin/kinesin dataset, type `help(example.data)`.

3 Constructing Structure Ensemble for Protein Family

Comparing multiple structures of homologous proteins and carefully analysing large multiple sequence alignments can help identify patterns of sequence and structural conservation and highlight conserved interactions that are crucial for protein stability and function (**?**). Bio3d and R provide a useful framework for such studies and facilitate the integration of sequence, structure and dynamics data in the analysis of protein evolution.

3.1 Find Available Sets of Similar Structures

In this tutorial, to collect available transducin crystal structures, we first use BLAST to query the PDB database to find similar sequences (and hence structures) to our chosen representative (PDB id 1tag):

```
pdb <- read.pdb("1tag")

##   Note: Accessing online PDB file
##   HEADER    GTP-BINDING PROTEIN                     23-NOV-94   1TAG

seq <- pdbseq(pdb)
blast <- blast.pdb(seq)

##  Searching ... please wait (updates every 5 seconds) RID = 85TE3J79014
##  .
##  Reporting 240 hits
```

Examining the alignment scores and their associated E-values (with the function `plot.blast()`) indicates a sensible normalized score (-log(E-Value)) cutoff of 240 bits (Figure 2).

```
hits <- plot.blast(blast, cutoff = 240)

##   * Possible cutoff values include:
##   709 239 -2
##     Yielding Nhits:
##   15 97 240
```

```
head(hits$hits)

##   pdb.id   gi.id    group
## 1 "1TND_A" "576308"  "1"
## 2 "1TND_B" "576309"  "1"
## 3 "1TND_C" "576310"  "1"
```

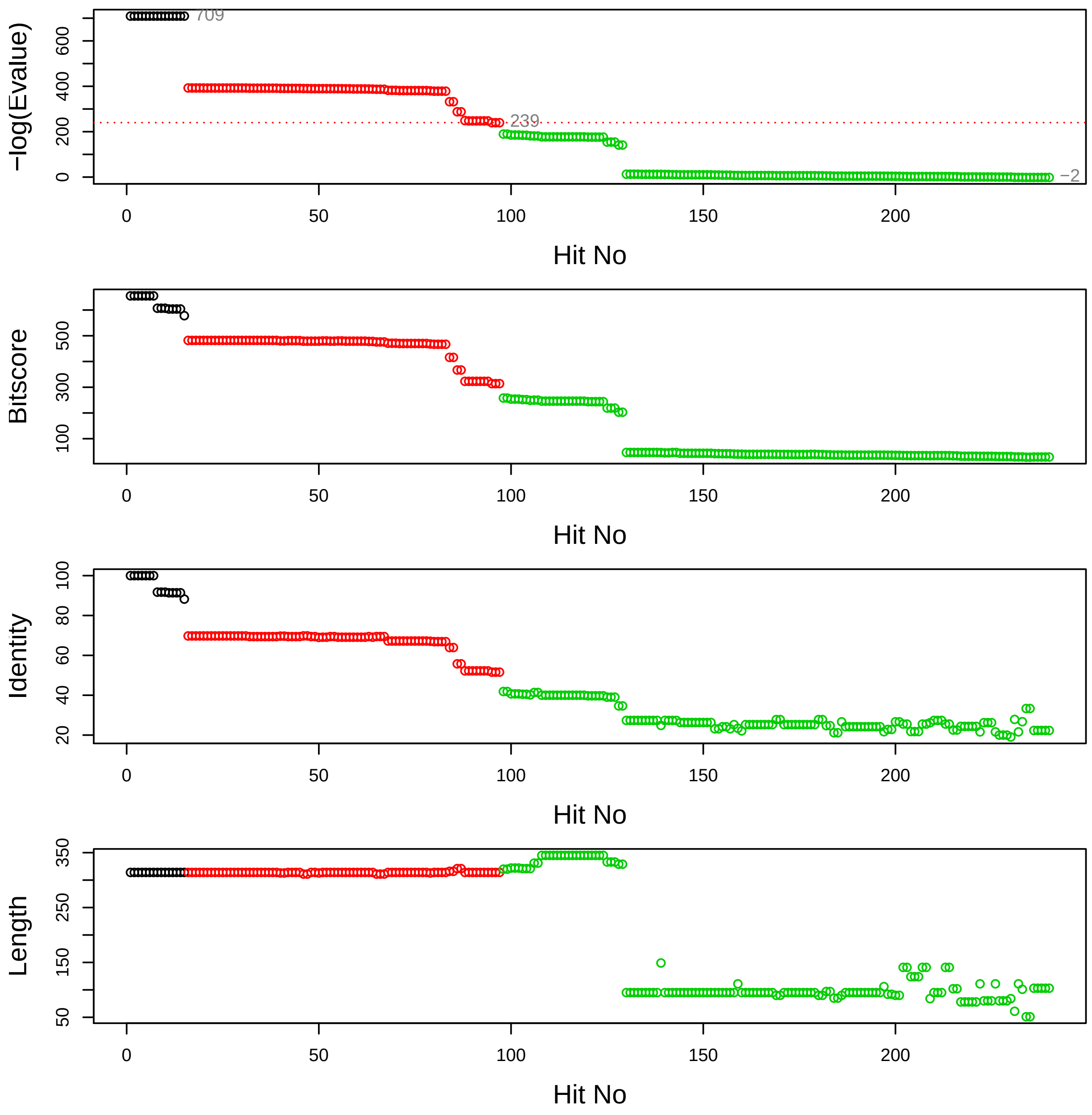


Figure 2: BLAST against PDB database with query sequence PDB id 1tag

```
## 4 "1TAD_A" "1065261" "1"
## 5 "1TAD_B" "1065262" "1"
## 6 "1TAD_C" "1065263" "1"
```

Side-note: The funciton `pdb.annotate()` can help to fetch more detailed information about the corresponding structures, e.g. title, experimental method, resolution, ligand name, citation, etc., providing the PDB ids:

```
anno <- pdb.annotate(hits$pdb.id)

## Warning: ids should be standard 4 character PDB formart: trying first 4 char...

head(anno[, c("resolution", "ligandId", "citation")])

##      resolution                 ligandId                           citation
## 1AGR        2.8 ALF, CIT, GDP, MG, null          Tesmer et al. Cell (1997)
## 1AS0        2.0             GSP, MG, SO4     Raw et al. Biochemistry (1997)
## 1AS2        2.8                 GDP, PO4     Raw et al. Biochemistry (1997)
## 1AS3        2.4                 GDP, SO4     Raw et al. Biochemistry (1997)
## 1BH2        2.1                  GSP, MG  Posner et al. J.Biol.Chem. (1998)
## 1BOF        2.2             GDP, MG, SO4 Coleman et al. Biochemistry (1998)
```

3.2 Multiple Sequence Alignment

After downloading our complete list of structures from the PDB and splitting these into separate chains (see below) we extract the ATOM record sequence from each structure and align these to determine residue-residue correspondences.

```
unq.ids <- unique(substr(hits$pdb.id, 1, 4))
## - Download and chain split PDBs
raw.files <- get.pdb(unq.ids, path = "raw_pdbs")
files <- pdbsplit(raw.files, ids = hits$pdb.id, path = "raw_pdbs/split_chain")
## - Extract and align sequences
pdbs <- pdbaln(files)
```

Inspect the alignment (the automatically generated "aln.fa" file) with your favorate alignment viewer (e.g. SEAVIEW, available from: http://pbil.univ-lyon1.fr/software/seaview.html).

Side-note: It is possible that you may find some structures with missing residues (or gaps in the alignment) at sites of particular interest (e.g. switch regions in the case of transducin). This will prevent you from investigating the structural variation about these important sites and so may cause misleading results in your studies. To solve this problem, you may exclude these "bad" sequences from your hit list and generate a smaller but with higher quality dataset for further exploration.

Question: How to generate a `pdbs` object with a subset hit PDB ids?

4 Comparative Structure Analysis

The detailed comparison of homologous protein structures can be used to infer pathways for evolutionary adaptation and, at closer evolutionary distances, mechanisms for conformational change. The bio3d package employs refined structural superposition and principal component analysis (PCA) to examine the relationship between different conformers.

4.1 Structure Superposition

Conventional structural superposition of proteins minimizes the root mean square difference between their full set of equivalent residues. However, for the current application such a superposition procedure can be inappropriate. For example, in the comparison of a multi-domain protein that has undergone a hinge-like rearrangement of its domains, standard all atom superposition would result in an underestimate of the true atomic displacement by attempting superposition over all domains (whole structure superposition). A more appropriate and insightful superposition would be anchored at the most invariant region and hence more clearly highlight the domain rearrangement (sub-structure superposition).

The `core.find()` function implements an iterated superposition procedure, where residues displaying the largest positional differences are identified and excluded at each round. The function returns an ordered list of excluded residues, from which the user can select a subset of 'core' residues upon which superposition can be based.

```
core <- core.find(pdbs)
```

There are `plot.core()` and `print.core()` functions available for further examining the output of `core.find()` (Figure 3).

```
col = rep("black", length(core$volume))
col[core$volume < 2] = "pink"
col[core$volume < 1] = "red"
plot(core, col = col)
```

The `print.core()` function also returns **atom** and **xyz** indices. Below we select a subset of positions with a cumulative ellipsoid volume of less than 1.0 Å^3 and use the returned indices to write a quick PDB file for viewing in a molecular graphics program (Figure 4).

```
inds <- print(core, vol = 1)

## # 88 positions (cumulative volume <= 1 Angstrom^3)
##    start end length
## 1     32  52     21
## 2    195 195      1
## 3    216 226     11
## 4    239 239      1
## 5    242 247      6
## 6    260 274     15
## 7    279 279      1
## 8    282 283      2
## 9    295 304     10
## 10   317 336     20

write.pdb(xyz = pdbs$xyz[1, inds$xyz], resno = pdbs$resno[1, inds$atom], file = "quick_core.pdb")
```

We can now superpose all structures on the selected core indices with the `fit.xyz()` function.

```
xyz <- fit.xyz(fixed = pdbs$xyz[1, ], mobile = pdbs, fixed.inds = inds$xyz,
    mobile.inds = inds$xyz)
```

The above command performs the actual superposition and stores the new coordinats in the matrix object `xyz`. By providing several extra arguments to `fit.xyz()` a directory, here named `fitlsq`, containing superposed structures can be produced.

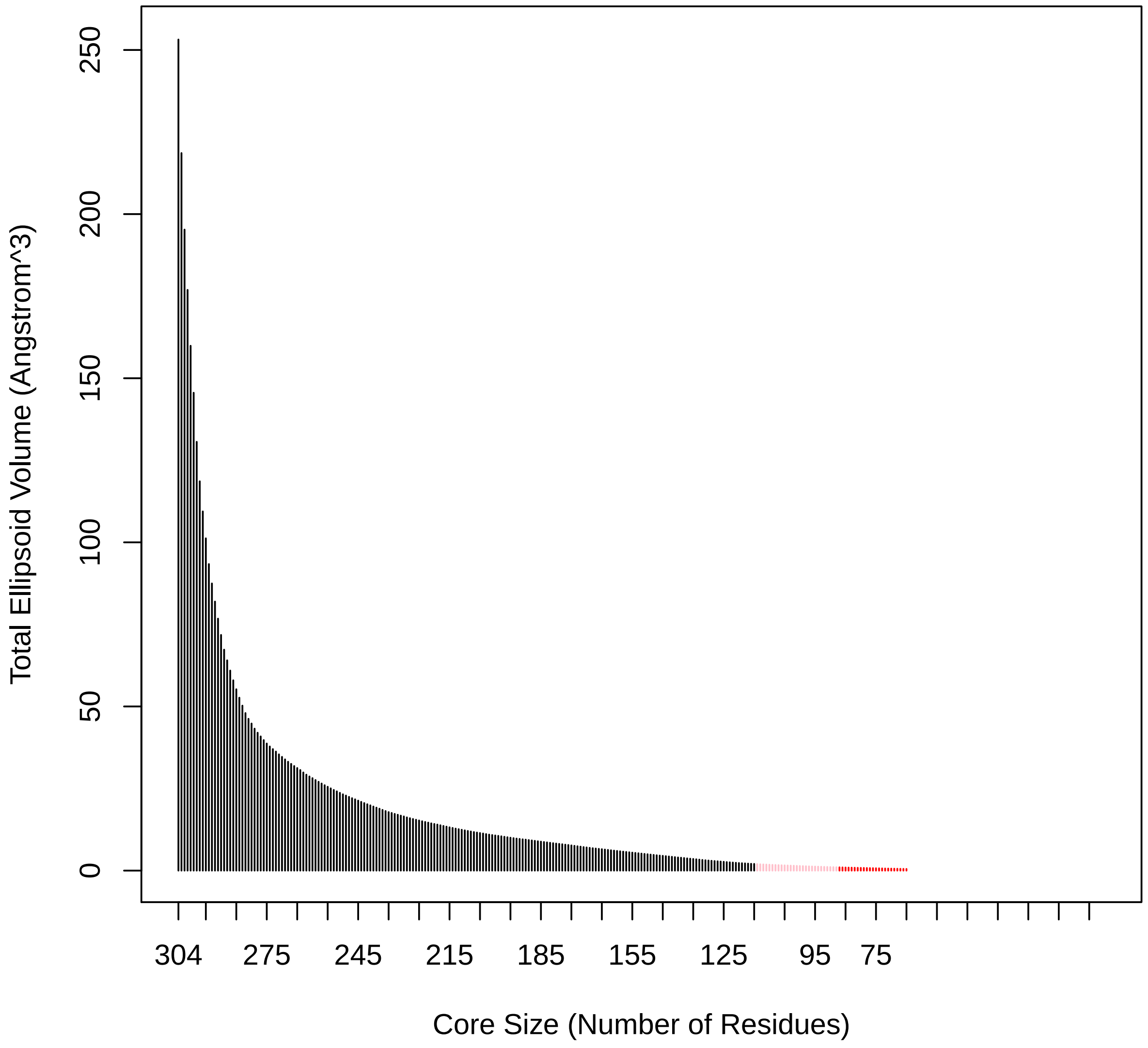


Figure 3: Identification of core residues

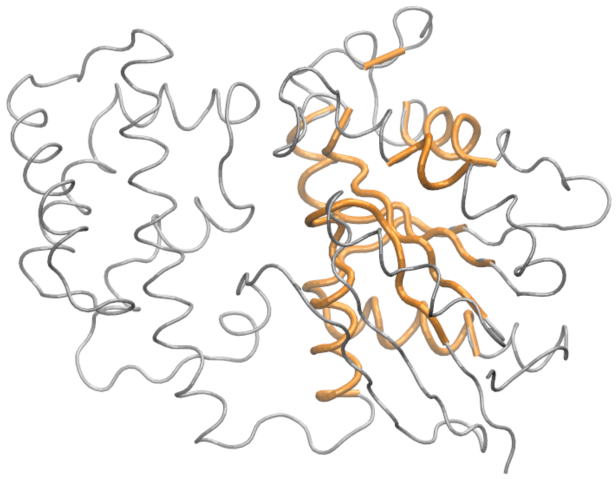


Figure 4: Core (the most structural invariant) positions in transducin family

```
xyz <- fit.xyz(fixed = pdbs$xyz[1, ], mobile = pdbs, fixed.inds = inds$xyz,
    mobile.inds = inds$xyz, prefix = files, pdbext = ".pdb", outpath = "fitlsq",
    full.pdbs = TRUE)
```

These can then be viewed in your favoriate molecular graphics program (Figure 5).

Question: Are the gap positions in `xyz` the same as those in `pdbs$xyz`?

4.2 Principal Component Analysis (PCA)

Following core identification and subsequent superposition, PCA can be employed to examine the relationship between different structures based on their equivalent residues. The application of PCA to both distributions of experimental structures and molecular dynamics trajectories, along with its ability to provide considerable insight into the nature of conformational differences has been discussed previously (see (**?**) and references therein).

Briefly, the resulting principal components (orthogonal eigenvectors) describe 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. Experience suggests that 3–5 dimensions are often sufficient to capture over 70 percent of the total variance in a given family of structures. Thus, a handful of principal components are sufficient to provide a useful description while still retaining most of the variance in the original distribution (**?**).

```
## Ignore gap containing positions
gaps.res <- gap.inspect(pdbs$ali)
gaps.pos <- gap.inspect(pdbs$xyz)
## -- Do PCA
pc.xray <- pca.xyz(xyz[, gaps.pos$f.inds])
```

Question: Why the input of `pca.xyz` is `xyz` rather than `pdbs$xyz`?

The above sequence of commands returns the indices of two structures and the indices for gap containing positions, both of which we exclude from subsequent PCA with the `pca.xyz()` command. A quick overview of the results of `pca.xyz()` can be obtained by calling `plot.pca()` (Figure 6).

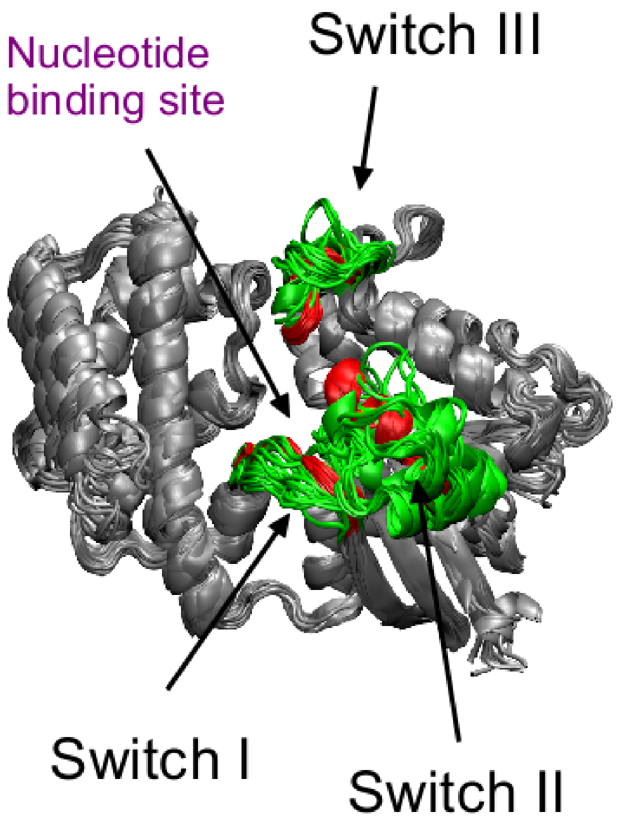


Figure 5: Structure ensemble of transducin family superposed based on core positions

```
plot(pc.xray, col = annotation[, "color"])
```

and calling `plot.bio3d()` to examine the contribution of each residue to the first three principal components (Figure 7).

```
## Plot loadings in relation to reference structure '1TAG'
sse <- dssp(pdb, resno = FALSE)
ind <- grep("1TAG", pdbs$id)
res.ref <- which(!is.gap(pdbs$ali[ind, ]))
res.ind <- which(res.ref %in% gaps.res$f.ind)
op <- par(no.readonly = TRUE)
par(mfrow = c(3, 1), cex = 0.6, mar = c(3, 4, 1, 1))
plot.bio3d(res.ind, pc.xray$au[, 1], sse = sse, ylab = "PC1 (A)")
plot.bio3d(res.ind, pc.xray$au[, 2], sse = sse, ylab = "PC2 (A)")
plot.bio3d(res.ind, pc.xray$au[, 3], sse = sse, ylab = "PC3 (A)")
```

```
par(op)
```

The plots in figures 6 and 7 display the relationships between different conformers, highlight positions responsible for the major differences between structures and enable the interpretation and characterization of multiple interconformer relationships.

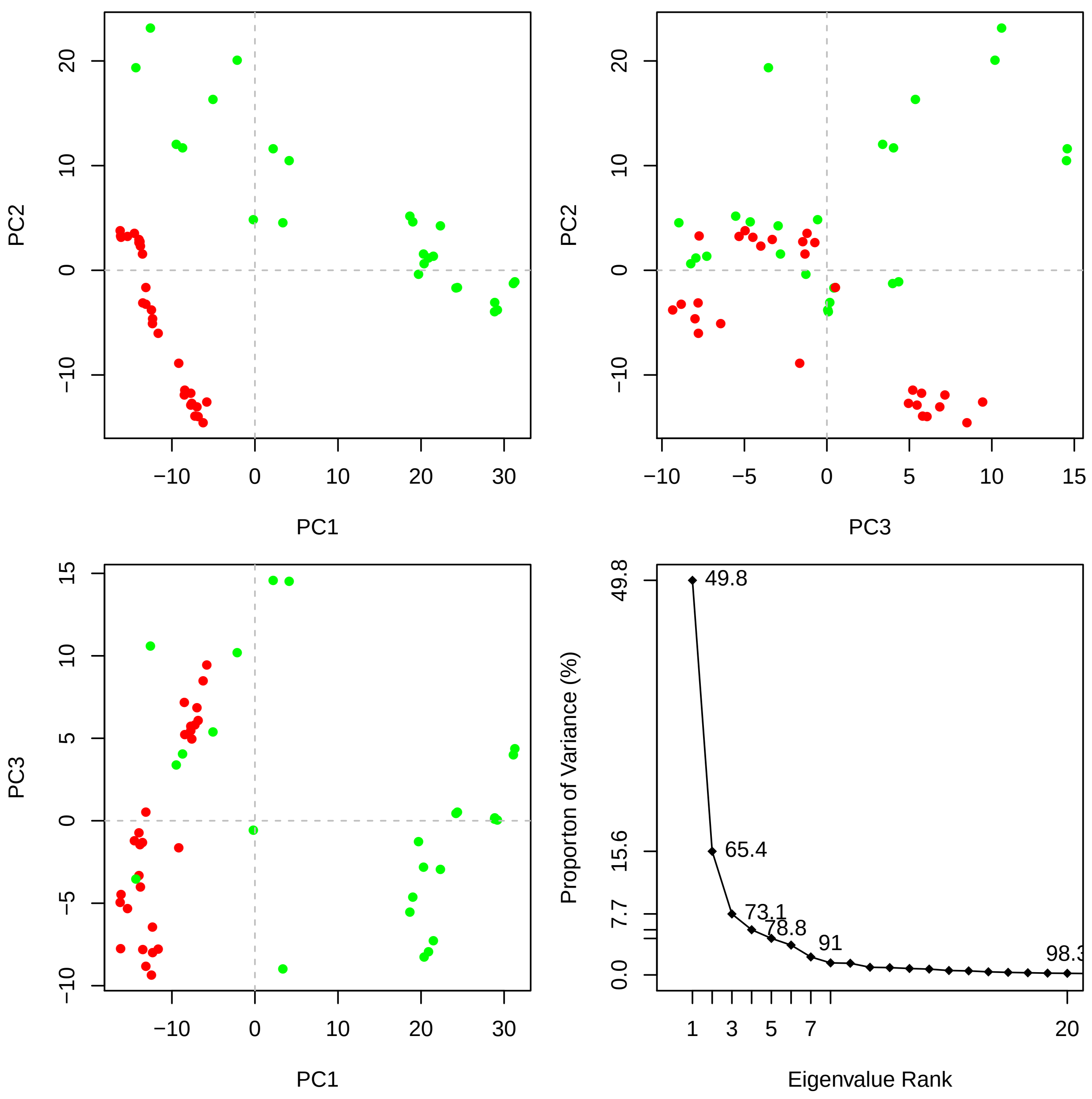


Figure 6: PCA of transducin X-ray structures

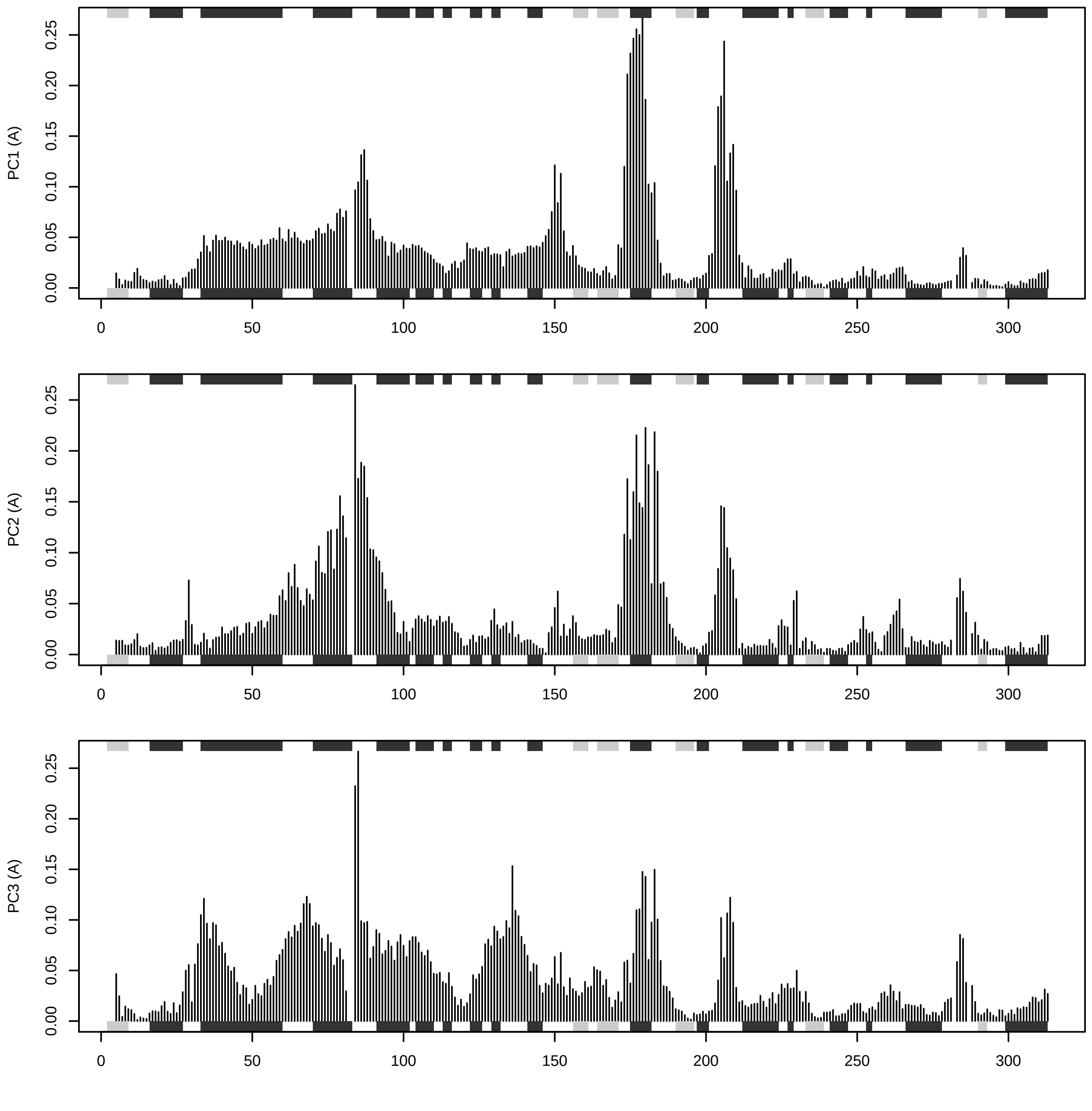


Figure 7: Contribution of each residue to the first three principal components

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 cmponent. This involves dividing the difference between the conformers into a number of evenly spaced steps along the principal components, forming the frames of the trajectory. Such trajectories can be directly visualized in a molecular graphics program, such as VMD (?). Furthermore, the interpolated structures can be analyzed for possible domain and shear movements with the DynDom package (?), or used as initial seed structures for more advanced reaction path refinement methods such as Conjugate Peak Refinement (?).

```r
a <- mktrj.pca(pc.xray, pc = 1, file = "pc1.pdb", resno = pdbs$resno[1, gaps.res$f.inds],
    resid = aa123(pdbs$ali[1, gaps.res$f.inds]))
```

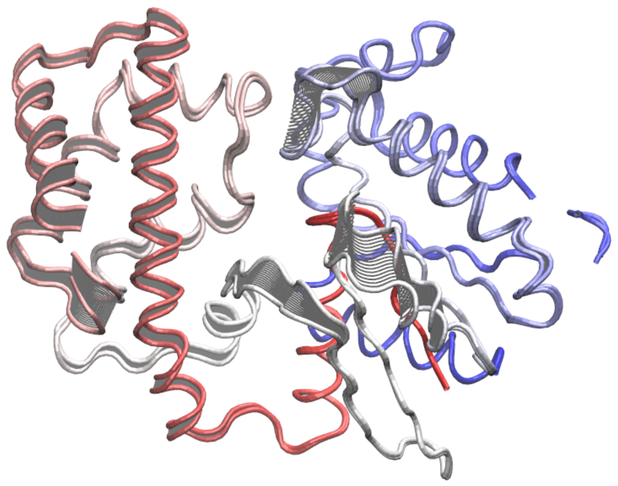


Figure 8: Interpolated structures along PC1

4.3 Conformer Clustering in PC Space

PCA provides a way of data representation with each dimension or principal component (PC) along the axis of maximum structural variance. Clustering of structures in the PC space enable us to focus on the major structural change in the dataset, with a controllable the level of dynamic details (via specifying the number of PCs used in the clustering). For example, with clustering in the 1 to 2 PCs, we can investigate how the X-ray structures of transducin relate to each other with respect to the major conformation change that covers over 65% structural variance (See above Figure 6).

```r
hc <- hclust(dist(pc.xray$z[, 1:2]))
grps <- cutree(hc, h = 30)
cols <- c("red", "green", "blue")
plot(pc.xray$z[, 1:2], typ = "p", pch = 16, cex = 1, col = cols[grps], xlab = "PC1",
    ylab = "PC2")
```

```r
plot(hc, labels = pdbs$id, main = "PC1-2", xlab = "", ylab = "Distance")
abline(h = 30, lty = 3, col = "gray60")
```

In the PC1-PC2 plane, the inactive ”GDP” state of structures (green points in Figure 6) are further split into two sub-groups (Figure 9 10). The bottom-right sub-group (blue) exclusively correspond to the structures complexed with GDP dissociation inhibitor (GDPi).

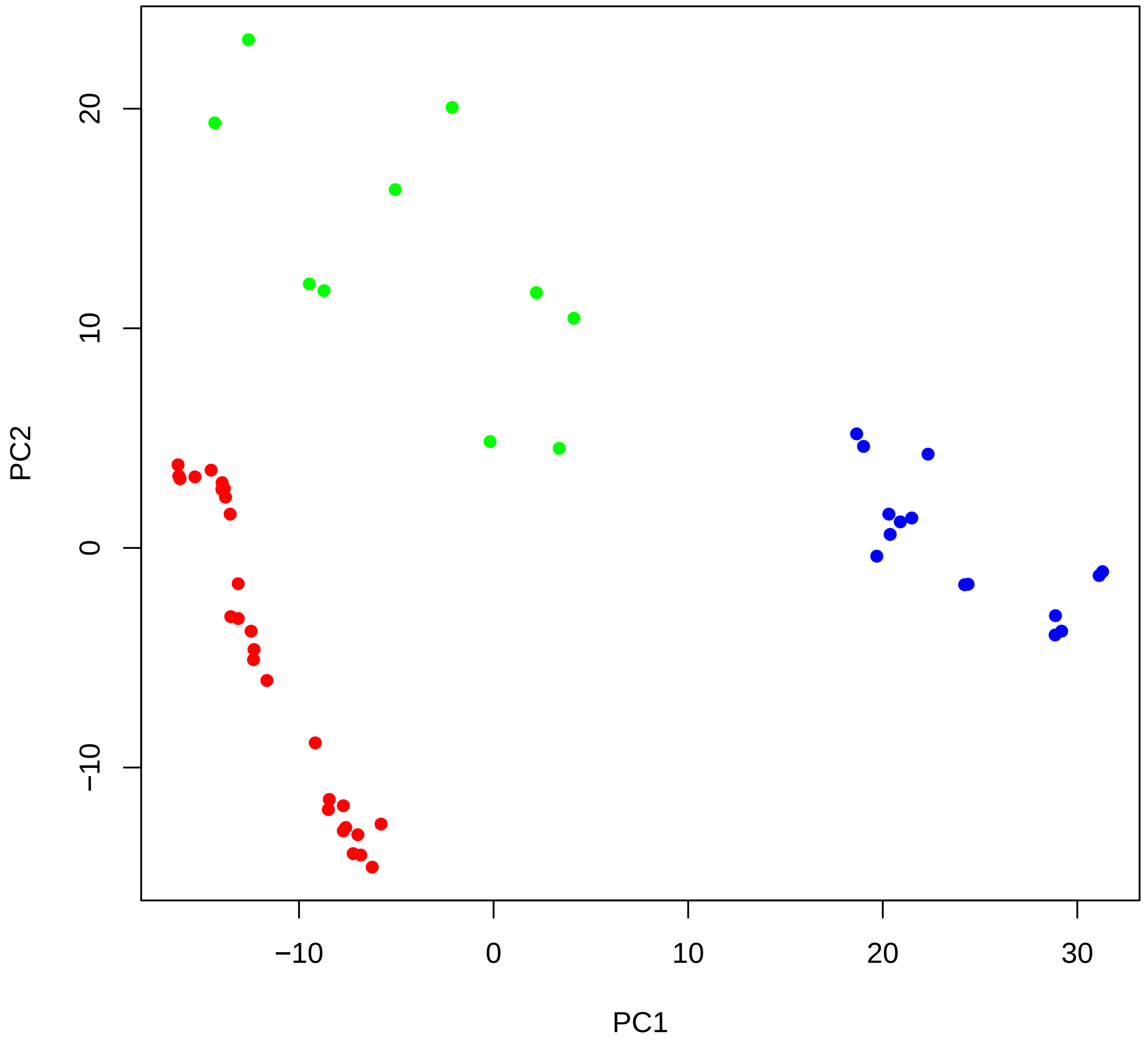


Figure 9: Clustering based on PC1-PC2

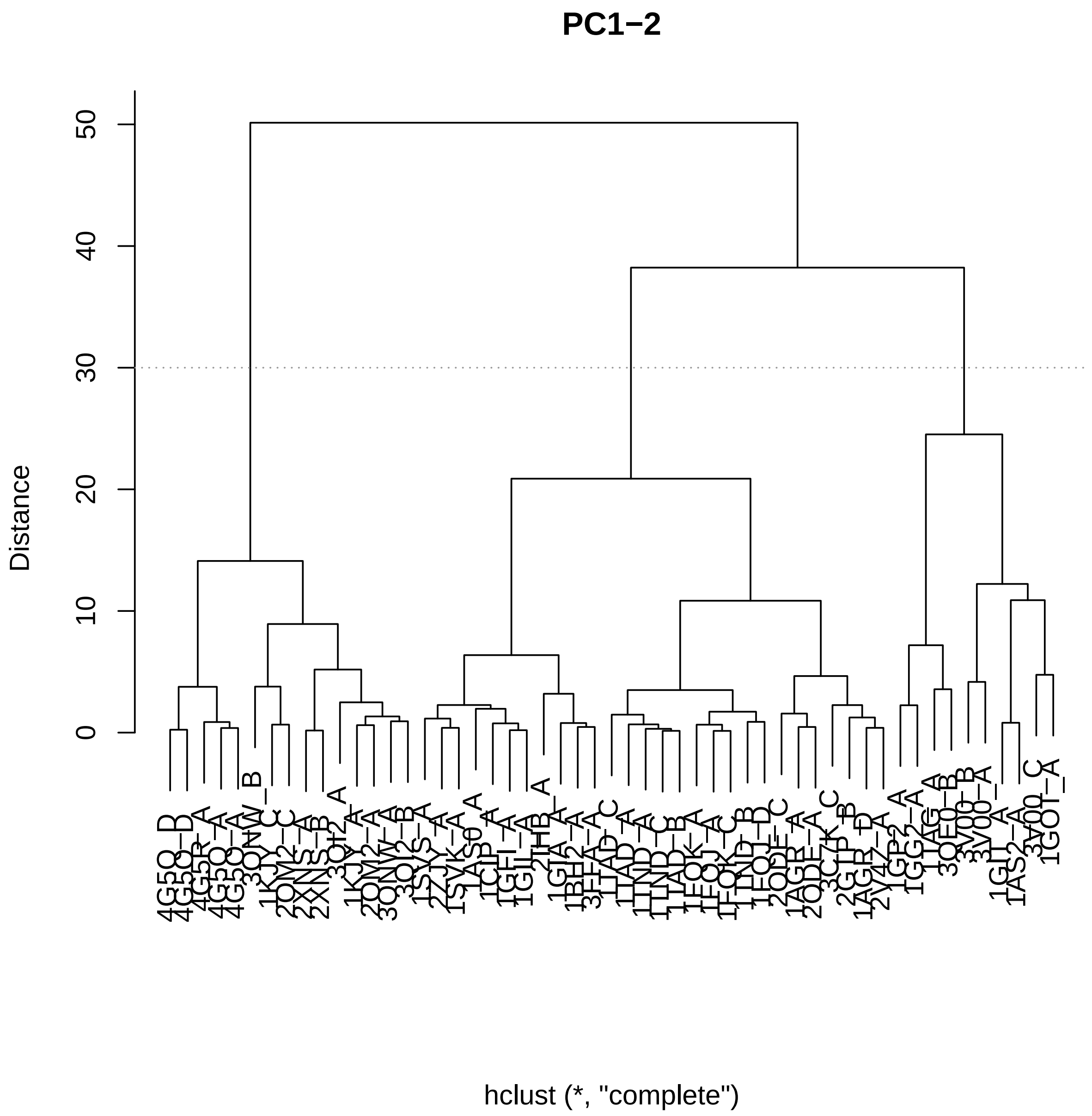


Figure 10: Clustering based on PC1-PC2

Question: What is the difference between the clustering method using PCs and that using RMSD?

4.4 More Structural Analysis

Bio3d facilitates the analysis of various structural properties, such as root mean-square deviation (RMSD), root mean-square fluctuation (RMSF), secondary structure, dihedral angles, difference distance matrices etc. The current section provides a brief exposure to using bio3d in this vein.

Root mean square deviation (RMSD): RMSD is a standard measure of structural distance between coordinate sets.

```
rmsd.fit <- rmsd(xyz[, gaps.pos$f.inds])
hist(rmsd.fit[upper.tri(rmsd.fit)], breaks = 20, xlab = "RMSD (A)")
```

Question: What about the RMSD distribution without fitting? What about fitting based on all Carbon-alpha positions?

Clustering based RMSD:

```
hc2 <- hclust(as.dist(rmsd.fit))
plot(hc2, labels = pdbs$id, ylab = "RMSD", xlab = "")
```

Root mean squared fluctuations (RMSF): RMSF is another often used measure of conformational variance. It usually returns a vector of atom-wise (or residue-wise) variance instead of a single numeric value.

```
rf <- rmsf(xyz[, gaps.pos$f.inds])
```

```
plot.bio3d(res.ind, rf, sse = sse, ylab = "RMSF (A)", xlab = "Position")
```

Torsion/Dihedral analysis and Difference distance matrix analysis (DDM): The conformation of a polypeptide or nucleotide chain can be usefully described in terms of angles of internal rotation around its constituent bonds.

```
tor <- torsion.pdb(pdb)
## basic Ramachandran plot
```

```
plot(tor$phi, tor$psi, xlab = "phi", ylab = "psi")
```

```
a.xyz <- pdbs$xyz["1TAG_A", ]
b.xyz <- pdbs$xyz["1TND_B", ]
gaps.xyz <- is.gap(pdbs$xyz["1TAG_A", ])
gaps.res <- is.gap(pdbs$ali["1TAG_A", ])
resno <- pdbs$resno["1TAG_A", !gaps.res]
a <- torsion.xyz(a.xyz[!gaps.xyz], atm.inc = 1)
b <- torsion.xyz(b.xyz[!gaps.xyz], atm.inc = 1)
d.ab <- wrap.tor(a - b)
a <- dm(a.xyz[!gaps.xyz])
```

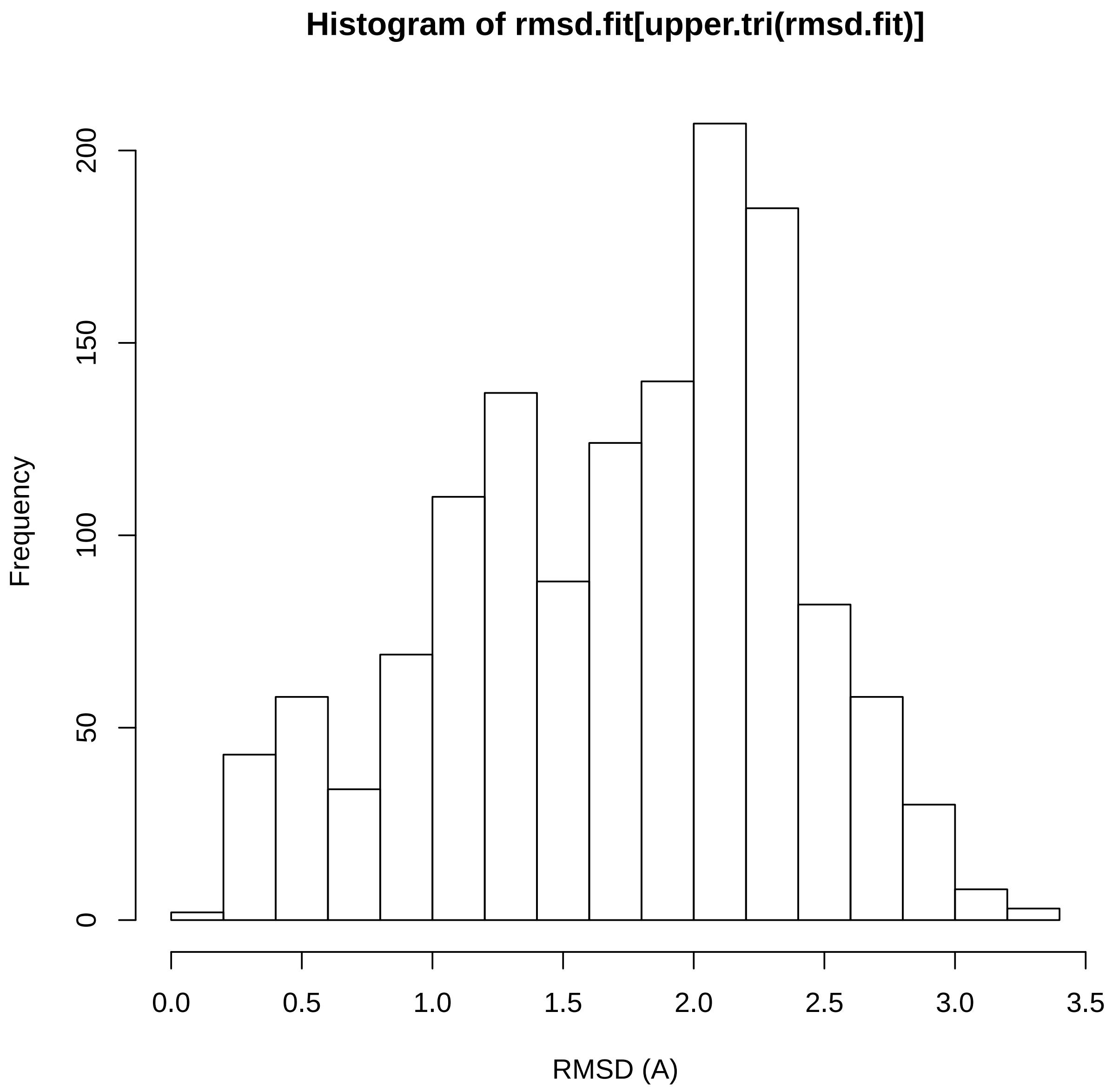


Figure 11: Histogram of RMSD among transducin structures

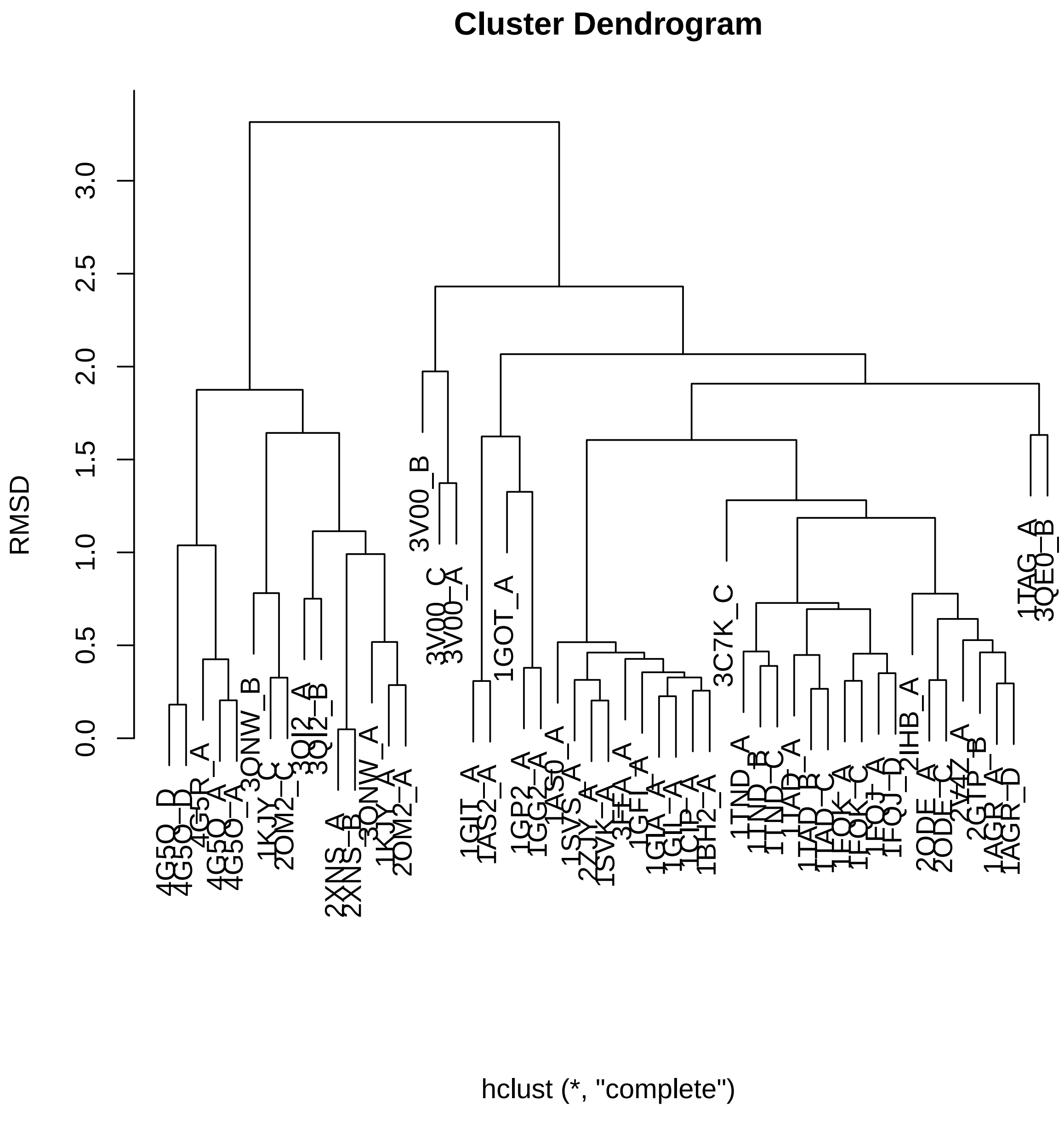


Figure 12: Clustering based on RMSD

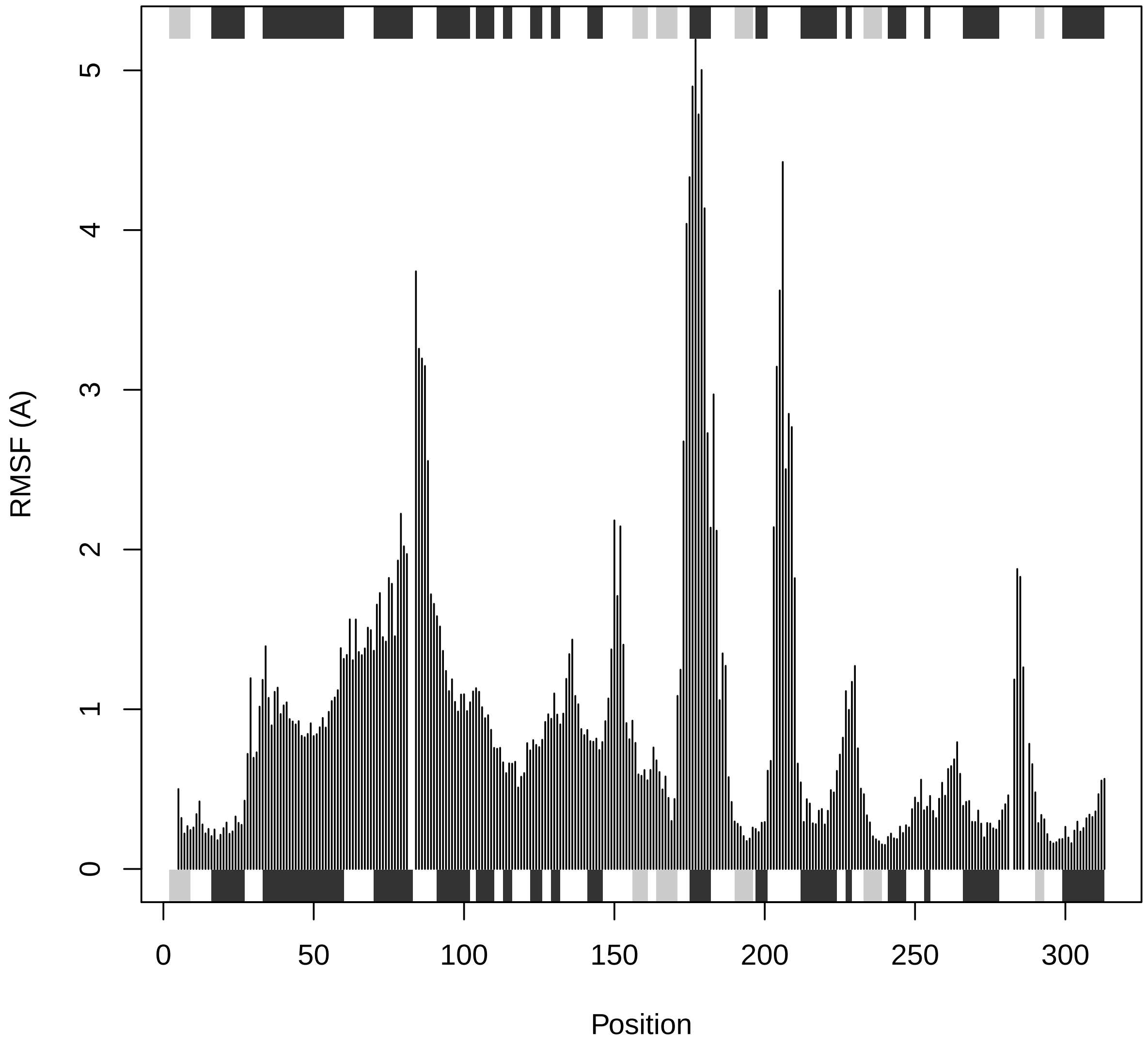


Figure 13: RMSF plot

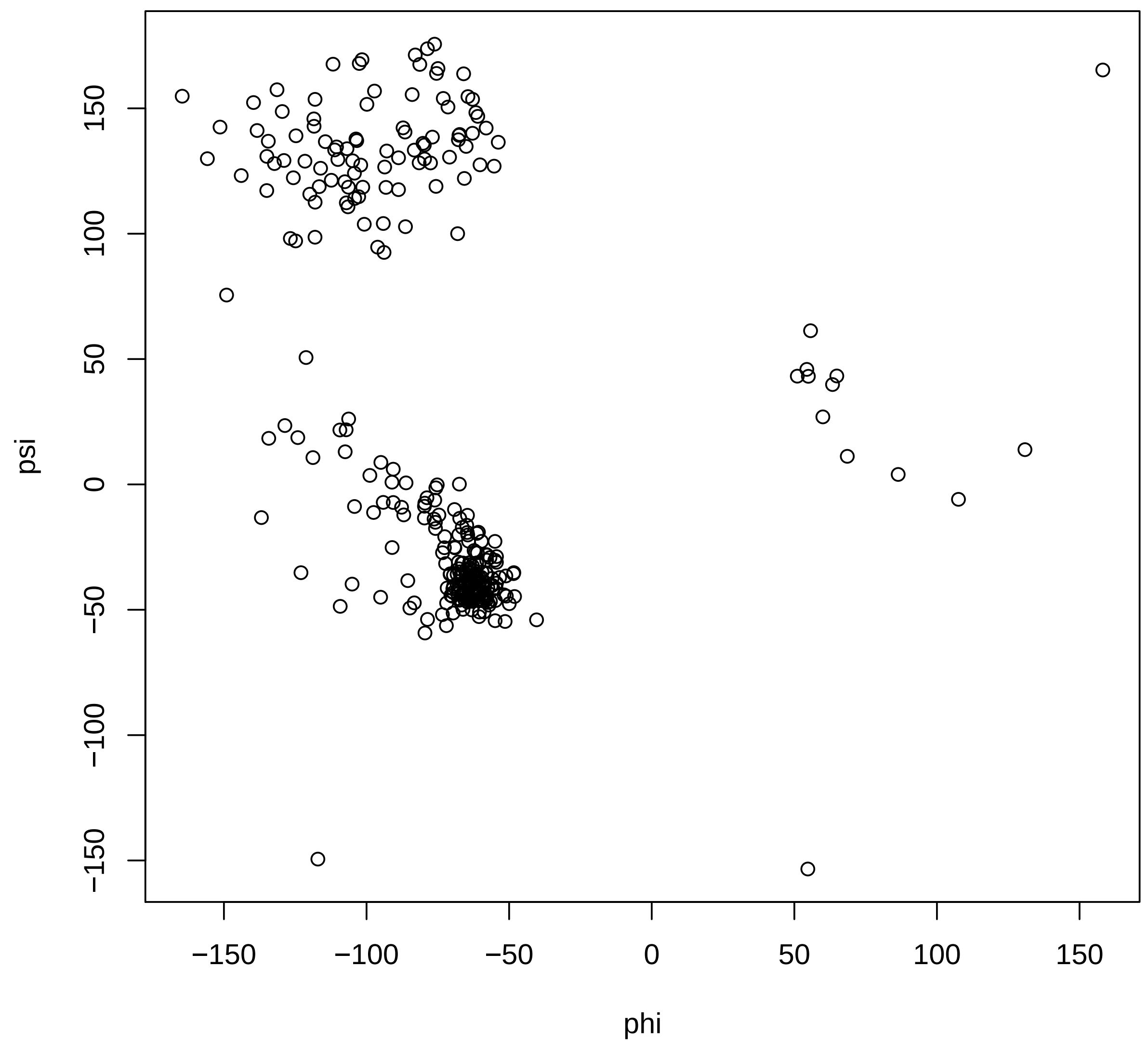


Figure 14: Ramachandran plot of the structure with PDB id 1tag

```
## input is raw 'xyz' thus 'selection' ignored

b <- dm(b.xyz[!gaps.xyz])

## input is raw 'xyz' thus 'selection' ignored

ddm <- a - b
```

```
sse2 <- dssp(read.pdb("1tag"))

##   Note: Accessing online PDB file
##   HEADER    GTP-BINDING PROTEIN                     23-NOV-94   1TAG

op <- par(no.readonly = TRUE)
par(mfrow = c(2, 1), mar = c(4, 4, 0, 1))
plot(resno, d.ab, typ = "h", xlab = "", ylab = "Angle")
plot.bio3d(resno, abs(d.ab), typ = "h", sse = sse2, xlab = "Residue", ylab = "Angle")
```

```
par(op)
```

```
plot(ddm, nlevels = 10, grid.col = "gray", resnum.1 = resno, resnum.2 = resno,
    xlab = "1tag", ylab = "1tnd (positions relative to 1tag)")
```

Question: What are the pros and cons of different methods: PCA, RMSD, RMSF, torsion, and distance matrix?

5 Sequence Conservation Analysis

In this section, we illustrate several functions related to sequence conservation analysis with the kinesin dataset. The `read.fasta` and `write.fasta` functions can be used to read and write aligned and non aligned sequences in FASTA format.

5.1 Sequence Alignment

The `seqaln()` function permits the alignment of multiple sequences as obtained from the `read.fasta()`. A simple alignment procedure for the sequences in the file *unaligned.fa* would involve the commands:

```
aln <- seqaln(read.fasta("unaligned.fa"))
```

5.2 Residue Conservation Analysis

To assess the level of sequence conservation at each position in an alignment, the *similarity*, *identity*, and *entropy* per position can be calculated with the `conserv()` function.

The *similarity* is defined as the average of the similarity scores of all pairwise residue comparisons for that position in the alignment, where the similarity score between any two residues is the score value between those residues in the chosen substitution matrix.

The *identity* i.e. the preference for a specific amino acid to be found at a certain position, is assessed by averaging the identity scores resulting from all possible pairwise comparisons at that position in the

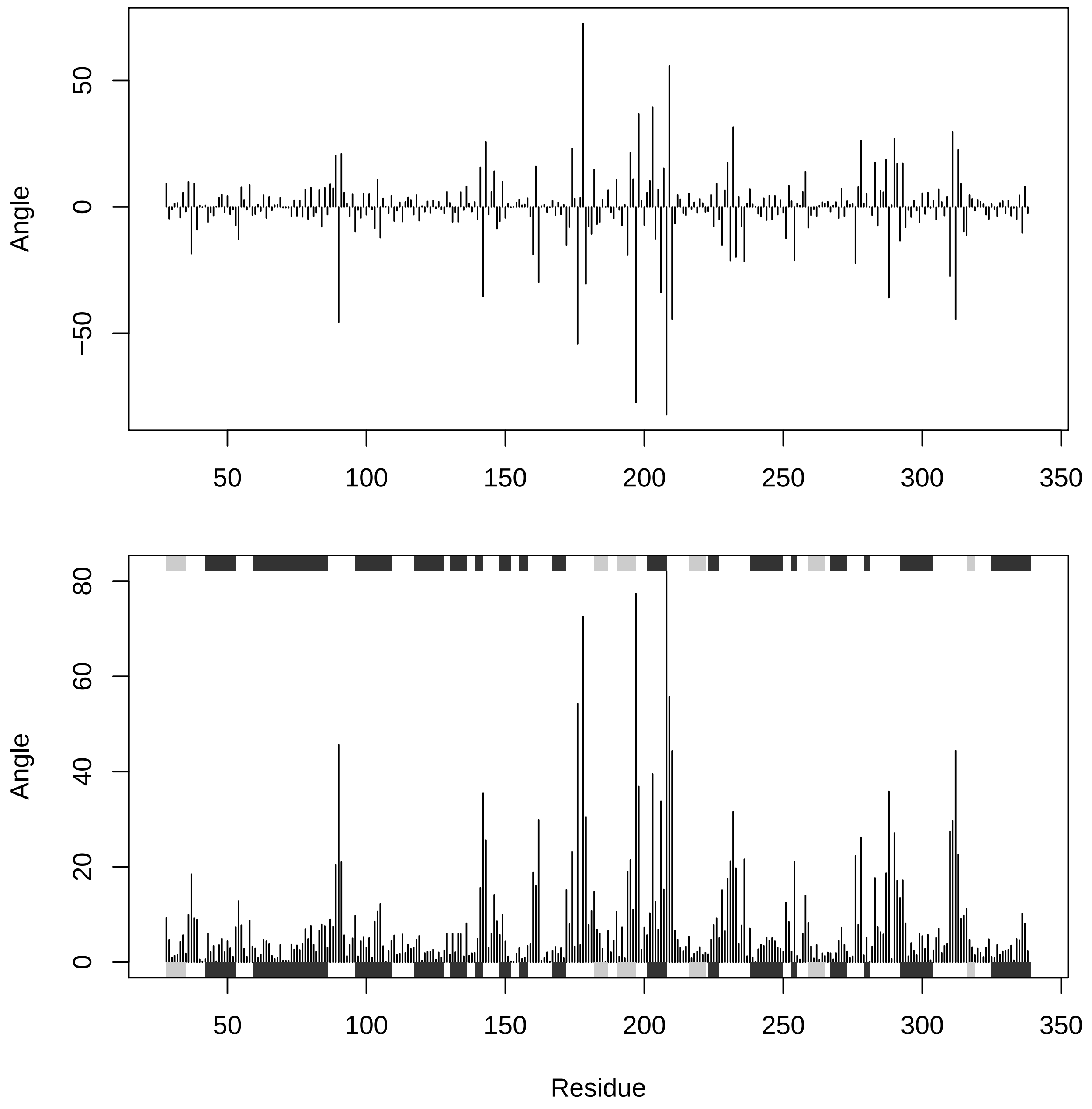


Figure 15: Torsion angle difference between structures in GDP (1tag) and GTP (1tnd) nucleotide states

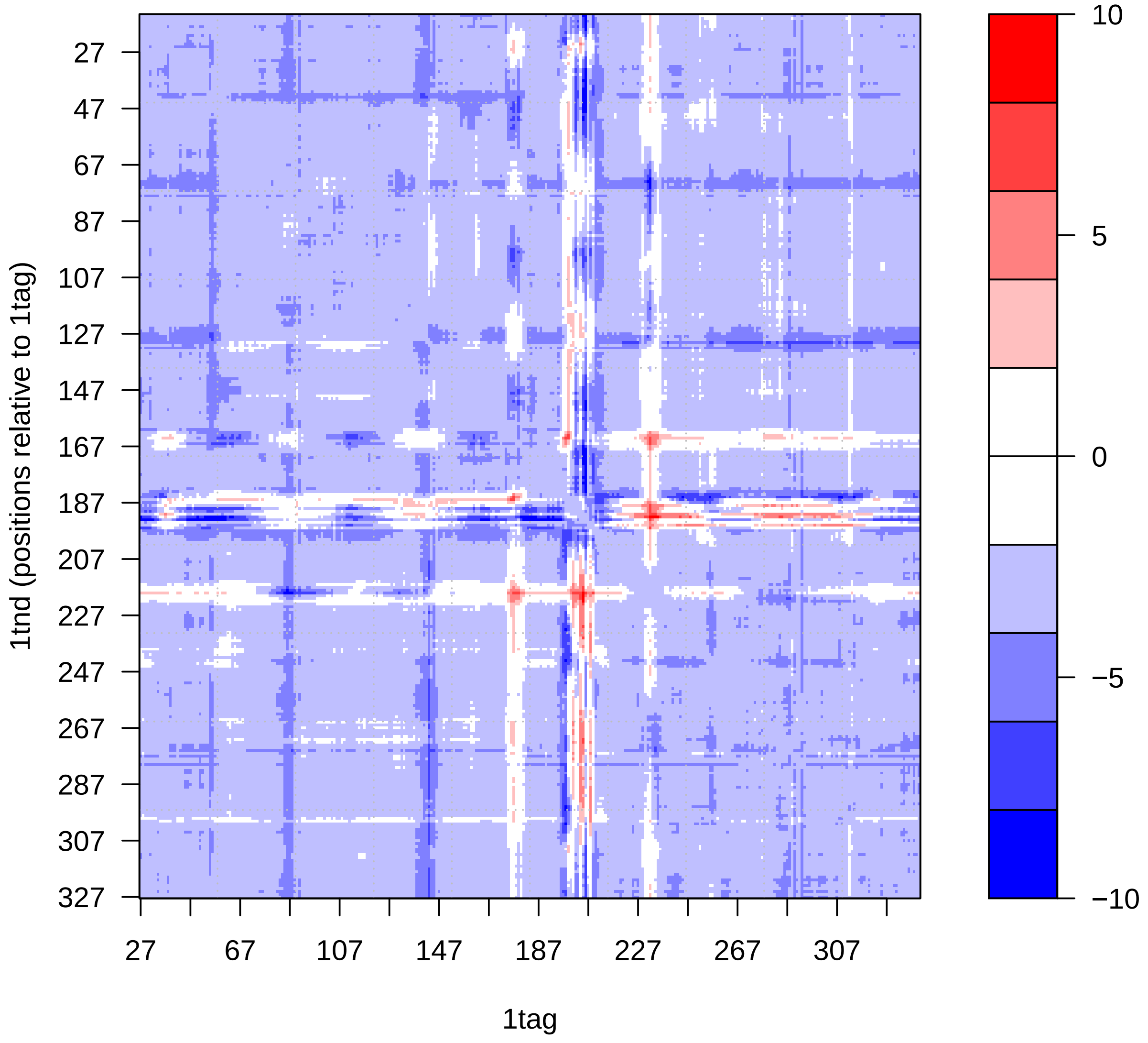


Figure 16: Difference of distance matrices between structures in GDP(1tag) and GTP(1tnd) nucleotide states

alignment, where all identical residue comparisons are given a score of 1 and all other comparisons are given a value of 0.

Entropy is based on Shannon's information entropy. See the `entropy` function for further details.

```
data(kinesin)
attach(kinesin, warn.conflicts = FALSE)
sim <- conserv(x = pdbs$ali, method = "similarity", sub.matrix = "bio3d")
write.fasta(pdbs, file = "kinesin.fa")
aln <- read.fasta("kinesin.fa")
```

Side-note: The last two lines in above code snippet are just for illustrating the usage of the function `read.fasta()`. In examples all through the document, the `aln` object can be replaced by `pdbs` because the latter contains all the components of the former.

```
pdb2 <- read.pdb("1bg2")

##   Note: Accessing online PDB file
##   HEADER    MOTOR PROTEIN                           04-JUN-98   1BG2

sse2 <- dssp(pdb2, resno = FALSE)
plot.bio3d(sim[!is.gap(aln$ali[1, ])], sse = sse2, xlab = "Residue", ylab = "Similarity")
```

The `aln2html()` function renders a sequence alignment as coloured HTML suitable for viewing with a web browser.

```
write.fasta(seqs = aln$ali[, 379:385], file = "eg.fa")
aln2html(aln, append = FALSE, file = "eg.html")
aln2html(aln, colorscheme = "ent", file = "eg.html")
```

Question: What is the relationship between sequence conservation and structural variance?

5.3 Inter-sequence Evolutionary Analysis

Pairwise identity analysis Pairwise identity analysis is an efficient way to remove too close sequences in the analysis, with the help of the funciton `ide.filter()`:

```
ide.mat <- seqidentity(pdbs)
# Histogram of pairwise identity values
op <- par(no.readonly = TRUE)
par(mfrow = c(2, 1))
hist(ide.mat[upper.tri(ide.mat)], breaks = 30, xlim = c(0, 1), main = "Sequence Identity",
    xlab = "Identity")
k <- ide.filter(ide = ide.mat, cutoff = 0.6)

## ide.filter(): N clusters @ cutoff =  10

ide.cut <- seqidentity(pdbs$ali[k$ind, ])
hist(ide.cut[upper.tri(ide.cut)], breaks = 10, xlim = c(0, 1), main = "Sequence Identity",
    xlab = "Identity")
```

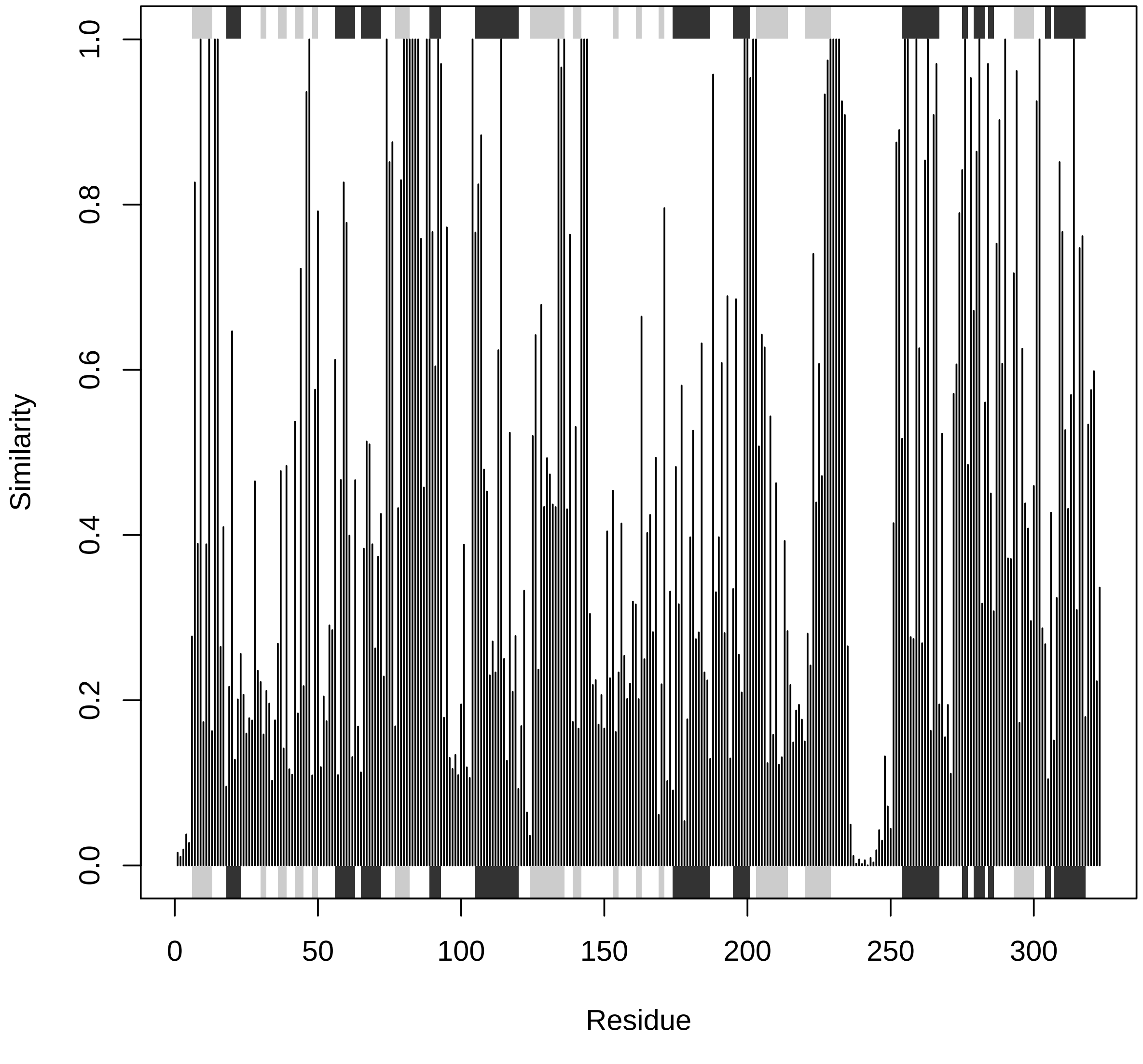
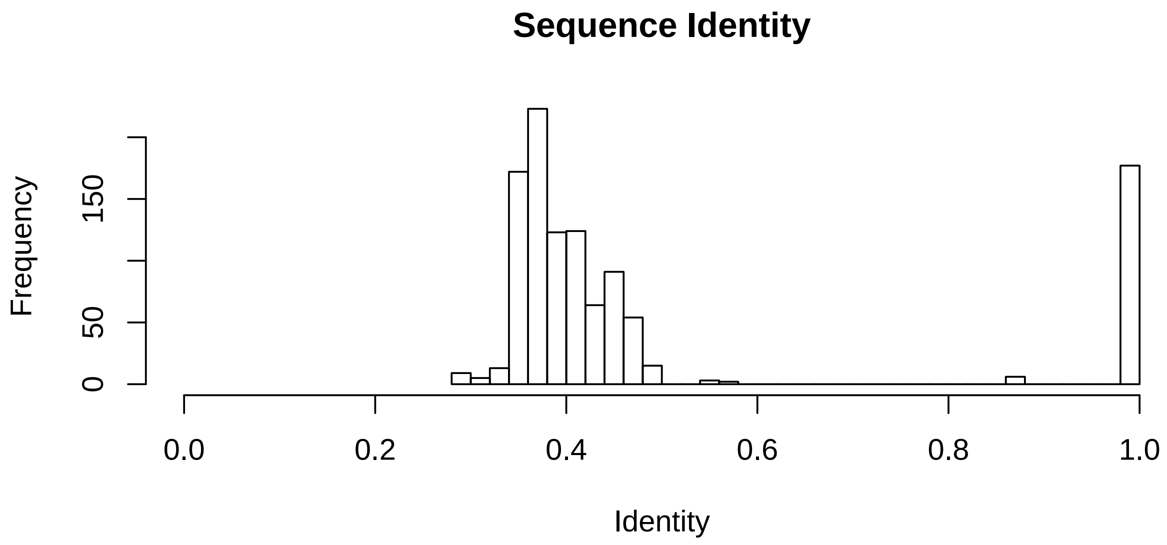


Figure 17: Residue conservation of kinesin protein family



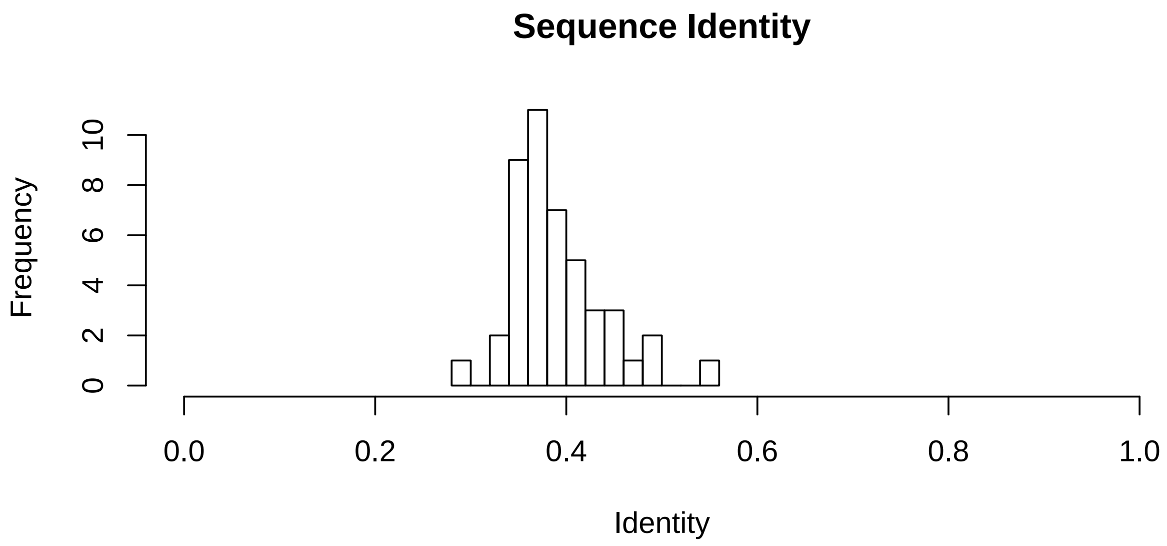


Figure 18: Sequence identity distribution before and after filter (cutoff=0.6)

```
par(op)
```

Consensus sequence Determines the consensus sequence for a given alignment at a given identity cutoff value. For clarity, we take the positions from 50 to 100 of the protein:

```
con <- consensus(aln$ali[, 50:100])
print(con$seq)

##  [1] "-" "-" "-" "-" "-" "-" "-" "I" "-" "V" "-" "-" "R" "-" "R" "P" "-"
## [18] "N" "-" "-" "E" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "I" "-"
## [35] "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "V" "-"

# Plot residue frequency matrix
col <- mono.colors(32)
aa <- rev(rownames(con$freq))
image(x = 1:ncol(con$freq), y = 1:nrow(con$freq), z = as.matrix(rev(as.data.frame(t(con$freq)))),
    col = col, yaxt = "n", xaxt = "n", xlab = "Alignment Position", ylab = "Residue Type")
# Add consensus along the axis
axis(side = 1, at = seq(0, length(con$seq), by = 5), labels = seq(50, 100, by = 5))
axis(side = 2, at = c(1:22), labels = aa)
axis(side = 3, at = c(1:length(con$seq)), labels = con$seq)
axis(side = 4, at = c(1:22), labels = aa)
grid(length(con$seq), length(aa))
box()
# Add consensus sequence
for (i in 1:length(con$seq)) {
    text(i, which(aa == con$seq[i]), con$seq[i], col = "white")
}
# Add lines for residue type separation
abline(h = c(2.5, 3.5, 4.5, 5.5, 3.5, 7.5, 9.5, 12.5, 14.5, 16.5, 19.5), col = "gray")
```

Question: What can we learn from a combined comparative sequence and structure analysis?

Session Info

```
toLatex(sessionInfo())

## \begin{itemize}\raggedright
##   \item R version 3.0.1 (2013-05-16), \verb|x86_64-redhat-linux-gnu|
##   \item Locale: \verb|LC_CTYPE=en_US.UTF-8|, \verb|LC_NUMERIC=C|, \verb|LC_TIME=en_US.UTF-8|, \verb|I
##   \item Base packages: base, datasets, graphics, grDevices,
##     methods, stats, utils
##   \item Other packages: bio3d~2.0, knitr~1.5.10, XML~3.98-1.1
##   \item Loaded via a namespace (and not attached): evaluate~0.5.1,
##     formatR~0.10, highr~0.2.1, stringr~0.6.2, tools~3.0.1
## \end{itemize}
```

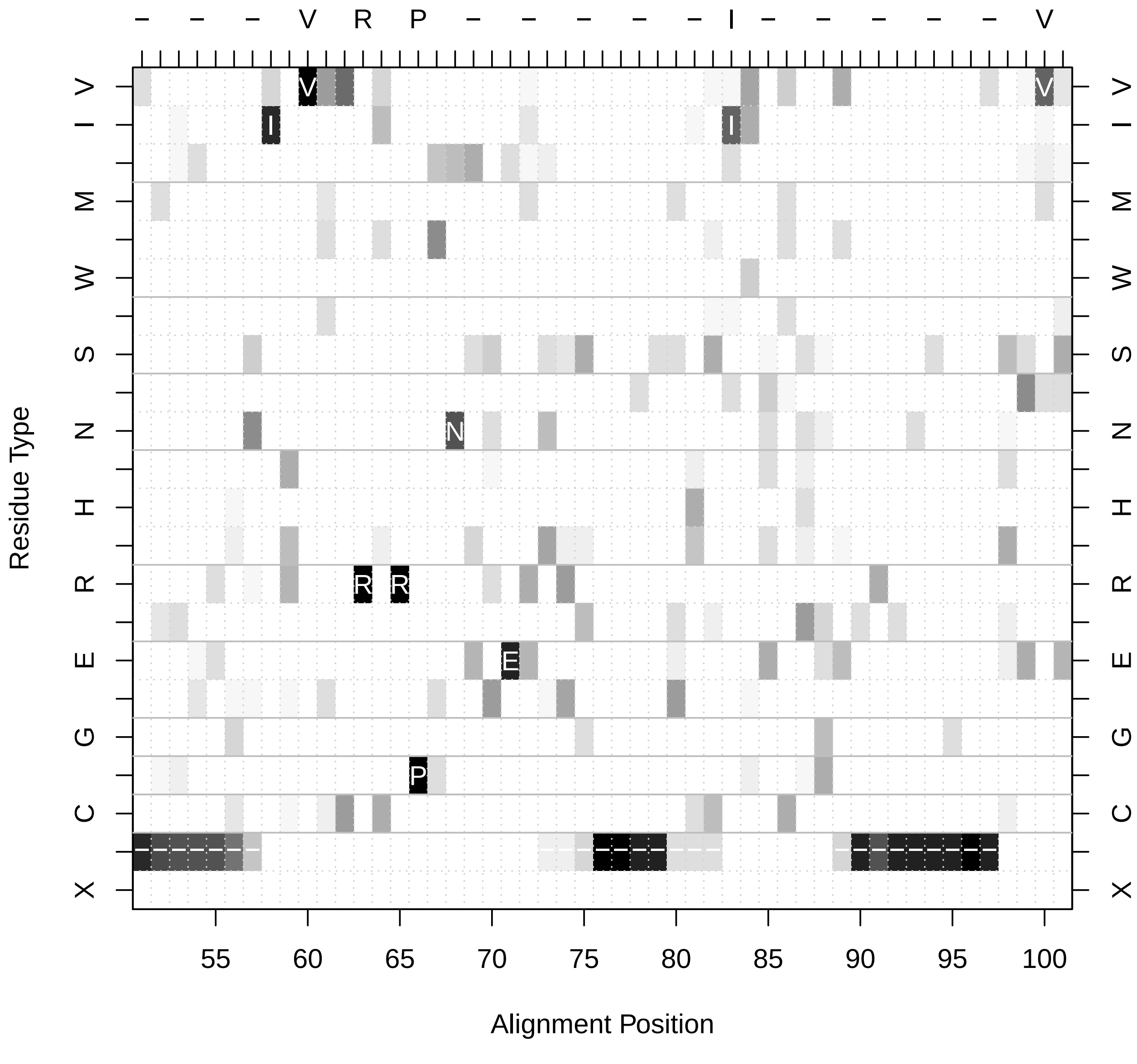


Figure 19: Consensus sequence identified for kinesin protein family

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.

Grant, B.J. and Mccammon, A.J. and Caves, L.S.D. and Cross, R.A. (2007) **Multivariate Analysis of Conserved Sequence-Structure Relationships in Kinesins: Coupling of the Active Site and a Tubulin-binding Sub-domain.** *J. Mol. Biol.*, **5**, 1231–1248

Fischer, S. and Karplus, M. (1992) **Conjugate peak refinement: an algorithm for finding reaction paths and accurate transition states in systems with many degrees of freedom.** *Chem. Phys. Lett*, **194**, 252–261

Hayward, S. and Berendsen, H. (1998) **Systematic analysis of domain motions in proteins from conformational change: new results on citrate synthase and T4 lysozyme.** *Proteins*, **30**, 144–154

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

Yao, X.Q. and Grant, B.J. (2013) **Domain-opening and dynamic coupling in the alpha-subunit of heterotrimeric G proteins.** *Biophys. J*, **105**, L08–10