Beginning Sequence and Structure Analysis with Bio3D

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

November 11, 2013

1 Introduction

The aim of this document, termed a vignette[1] in R parlance, is to provide a task-oriented introduction to the bio3d R package (Grant *et al.*, 2006). Bio3d contains utilities to process, organize and explore protein structure, sequence and dynamics data. Features include the ability to read and write structure, sequence and dynamic trajectory data, perform 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, various utility functions are provided to enable the statistical and graphical power of the R environment to work with biological sequence and structural data.

Supporting Material: The latest version and further documentation can be obtained from the bio3d website: http://thegrantlab.org/bio3d/ and wiki: http://bio3d.pbwiki.com/.

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"
 [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"
```

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

```
 [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"      "pdb2aln"           "pdb2aln.ind"
 [79] "pdbaln"            "pdbfit"            "pdbs2pdb"
 [82] "pdbseq"            "pdbsplit"          "plot.bio3d"
 [85] "plot.blast"        "plot.core"         "plot.dccm"
 [88] "plot.dccm2"        "plot.dmat"         "plot.enma"
 [91] "plot.nma"          "plot.pca"          "plot.pca.loadings"
 [94] "plot.pca.score"    "plot.pca.scree"    "plot.rmsip"
 [97] "print.core"        "print.enma"        "print.nma"
[100] "print.pdb"         "print.rle2"        "print.sse"
[103] "read.all"          "read.crd"          "read.dcd"
[106] "read.fasta"        "read.fasta.pdb"    "read.mol2"
[109] "read.ncdf"         "read.pdb"          "read.pdcBD"
[112] "read.pqr"          "rgyr"              "rle2"
[115] "rmsd"              "rmsd.filter"       "rmsf"
[118] "rmsip"             "rot.lsq"           "sdENM"
[121] "seq2aln"           "seqaln"            "seqaln.pair"
[124] "seqbind"           "seqidentity"       "sse.bridges"
[127] "store.atom"        "stride"            "struct.aln"
[130] "summary.pdb"       "torsion.pdb"       "torsion.xyz"
[133] "transducin"        "trim.pdb"          "unbound"
[136] "vec2resno"         "view.dccm"         "view.modes"
[139] "vmd.colors"        "wrap.tor"          "write.crd"
[142] "write.fasta"       "write.ncdf"        "write.pdb"
[145] "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 directly into your R session:

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

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

In the above example the function `read.pdb()` is supplied with one input argument, the four letter RCSB Protein Data Bank identifier `1tag`. As this In the above example the `get.pdb()` command returns the RCSB Protein Data Bank file for structure `1tag`. Setting `URLonly` to `FALSE` would cause the file to be downloaded to the current working directory. The `read.pdb()` command then processes this file and returns its output to the object `pdb`. To examine the contents of the `pdb` object we can use the `attributes` command.

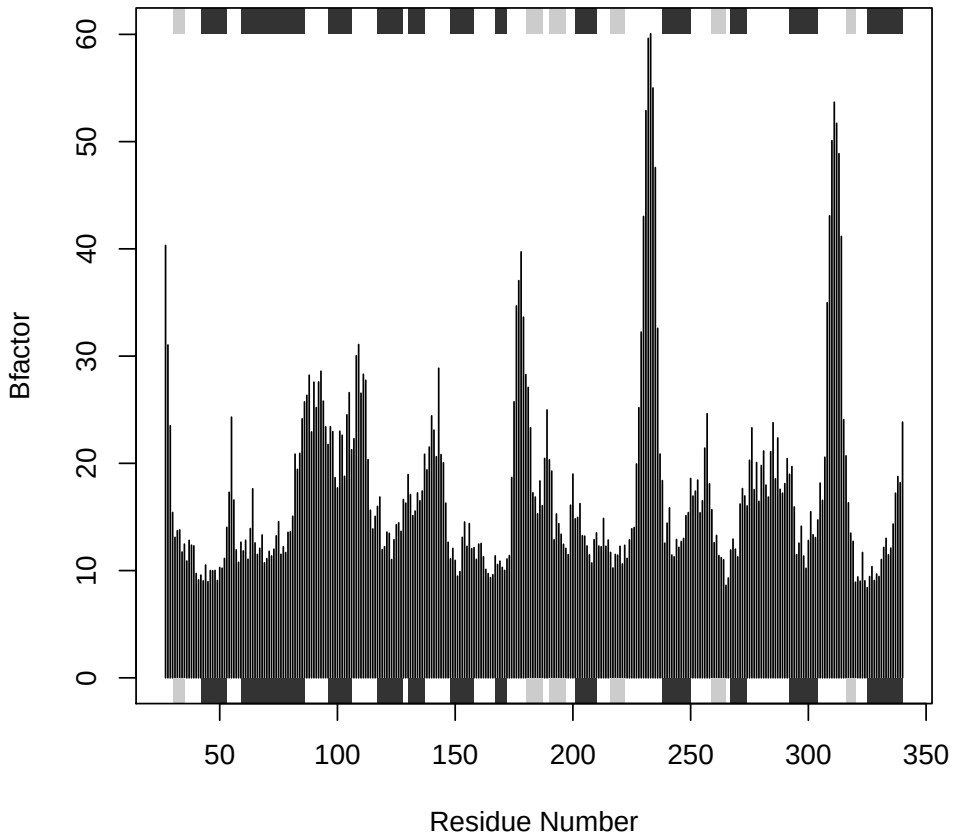
```
> attributes(pdb)

$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"
```

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 convinent 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="Resi
```



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. We will return to discuss atom selection in more detail in a subsequent section but as a further example of data access lets extract the sequence for the loop region between strand six and helix 11 (alpha 4) in our `pdb` object.

```
> loop <- pdb$sheet$end[6]:pdb$helix$start[11]
> loop.inds <- atom.select(pdb,paste("///", paste(loop, collapse=","), "///CA/", sep=""))

Build selection from input string

      segid  chain
Stest ""     ""
Natom "2521" "2521"
      resno
Stest "320,319,318,317,316,315,314,313,312,311,310,309,308,307,306,305,304,303,302,301,300,299,298,297,2
Natom "249"
      resid  eleno  elety
Stest ""     ""     "CA"
Natom "2521" "2521" "314"
 *  Selected a total of: 29 intersecting atoms  *

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

 [1] "TYR" "GLU" "ASP" "ALA" "GLY" "ASN" "TYR" "ILE" "LYS" "VAL" "GLN" "PHE"
[13] "LEU" "GLU" "LEU" "ASN" "MET" "ARG" "ARG" "ASP" "VAL" "LYS" "GLU" "ILE"
[25] "TYR" "SER" "HIS" "MET" "THR"

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

 [1] "Y" "E" "D" "A" "G" "N" "Y" "I" "K" "V" "Q" "F" "L" "E" "L" "N" "M" "R" "R"
[20] "D" "V" "K" "E" "I" "Y" "S" "H" "M" "T"
```

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 aminoacid codes.

Note that `pdb` object can be passed to other bio3d functions without the need for atom selection, including `torsion.pdb()`, `convert.pdb()`, `dssp()` and the distance matrix function `dm()`:

1.2 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.

1.2.1 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 (Yao and Grant, 2013). 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)
```

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 IDs:

```
> ## 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)
```

1.2.2 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 (Grant *et al.*, 2007). 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)
```

2 Sample Applications of bio3d

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 (Grant *et al.*, 2007). 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.

2.1 Comparative Structural 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.

2.1.1 Interconformer Relationships with PCA

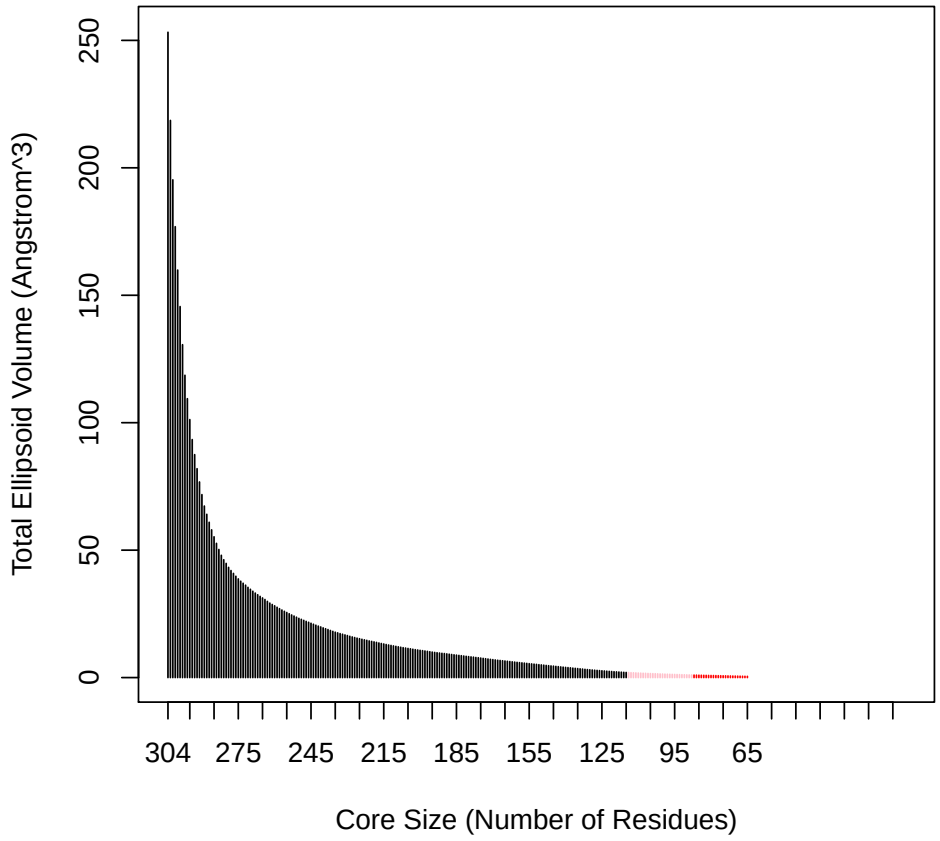
Conventional structural superposition of proteins minimizes the root mean square difference between their full set of equivalent residues. However, for the current application such a superposition procedure can be inappropriate. For example, in the comparison of a multi-domain protein that has undergone 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).

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()`.

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



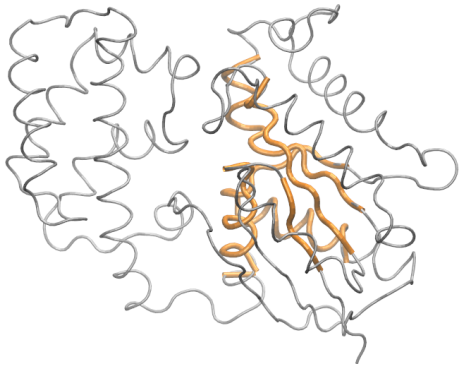
The `print.core()` function also returns `atom` and `xyz` indices. Below we select a subset of positions with a cumulative ellipsoid volume of less than 0.5 $Å^3$ and use the returned indices to write a quick PDB file for viewing in a molecular graphics program (Figure 2).

```
> inds <- print(core, vol=0.5)
```

```
# 66 positions (cumulative volume <= 0.5 Angstrom^3)
  start end length
1    32  36      5
2    39  49     11
3    51  52      2
```

```
4   195 195      1
5   216 226     11
6   260 274     15
7   299 299      1
8   317 336     20
```

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



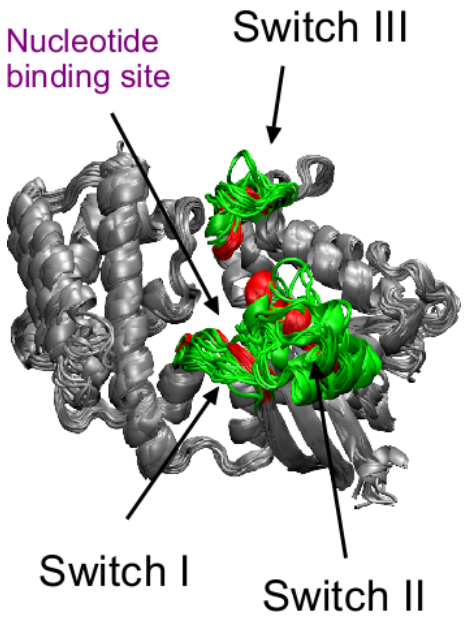
We can now superimpose all structures on the selected core indices with the `fit.xyz()` function.

```
> 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.

```
> 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 3).



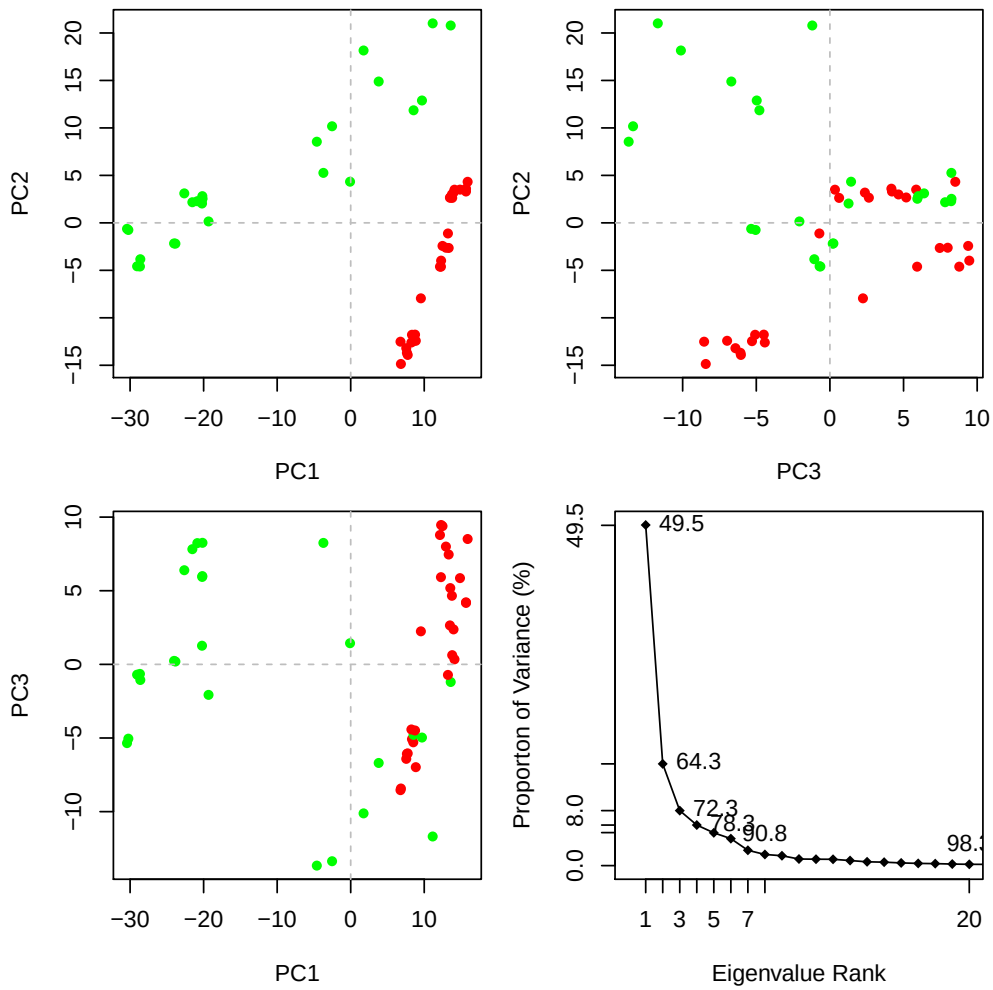
Principal Component Analysis: Following core identification and subsequent superposition, PCA can be employed to examine the relationship between different structures based on their equivalent residues. The application of PCA to both distributions of experimental structures and molecular dynamics trajectories, along with its ability to provide considerable insight into the nature of conformational differences has been discussed previously (see Grant *et al.* (2007) 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 Grant *et al.* (2006).

```
> ## 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])
```

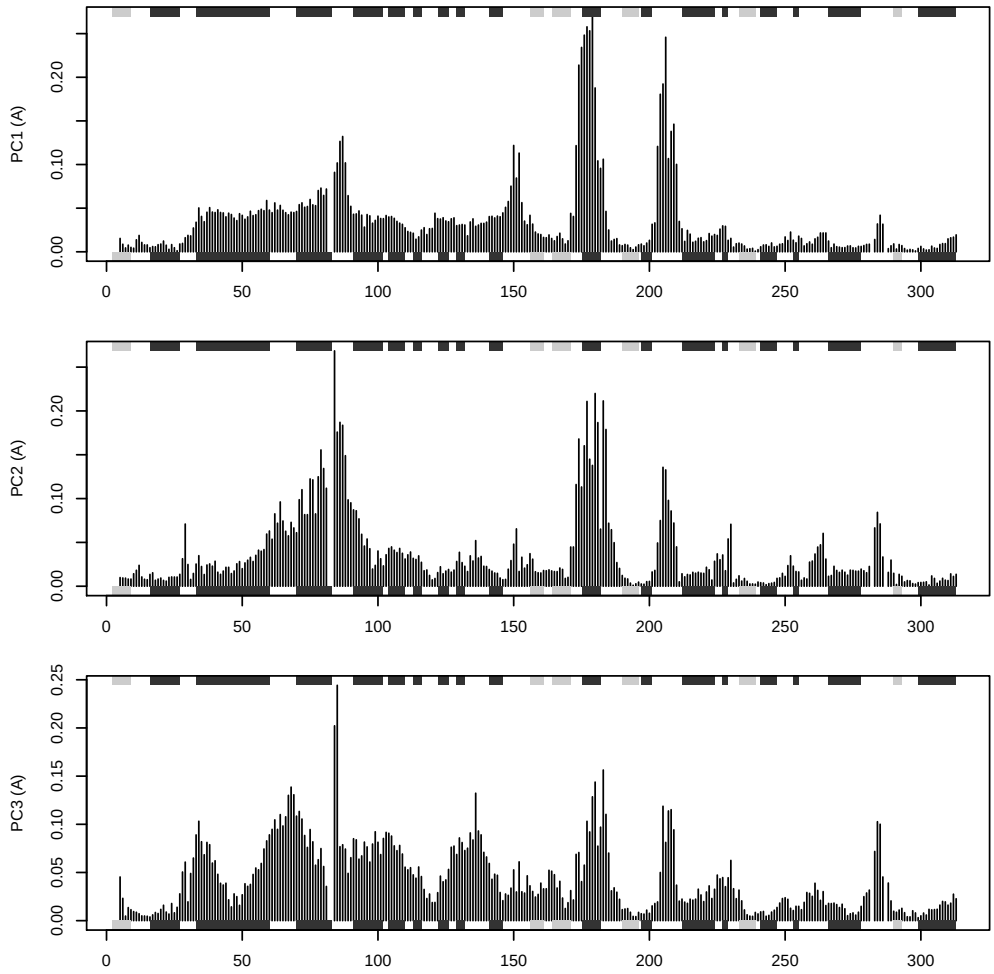
The above sequence of commands returns the indices of two structures and the indices for gap containing positions, both of which we exclude from subsequent PCA with the `pca.xyz()` command. A quick overview of the results of `pca.xyz()` can be obtained by calling `plot.pca()`

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



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

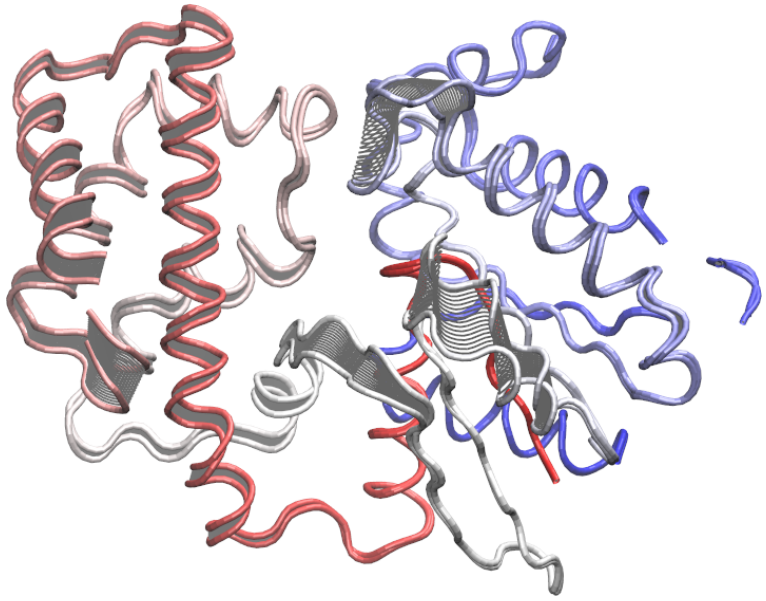
```
> ## Plot loadings in relation to reference structure "1TAG"
> sse <- dssp(pdb, resno=FALSE)
> ind <- grep("1TAG", pdbs$id)
> res.ref <- which(!is.gap(pdbs$ali[ind,]))
> res.ind <- which(res.ref %in% gaps.res$f.ind)
> op <- par(no.readonly=TRUE)
> par(mfrow = c(3, 1), cex = 0.6, mar = c(3, 4, 1, 1))
> plot.bio3d(res.ind, pc.xray$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 4 and 5 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.

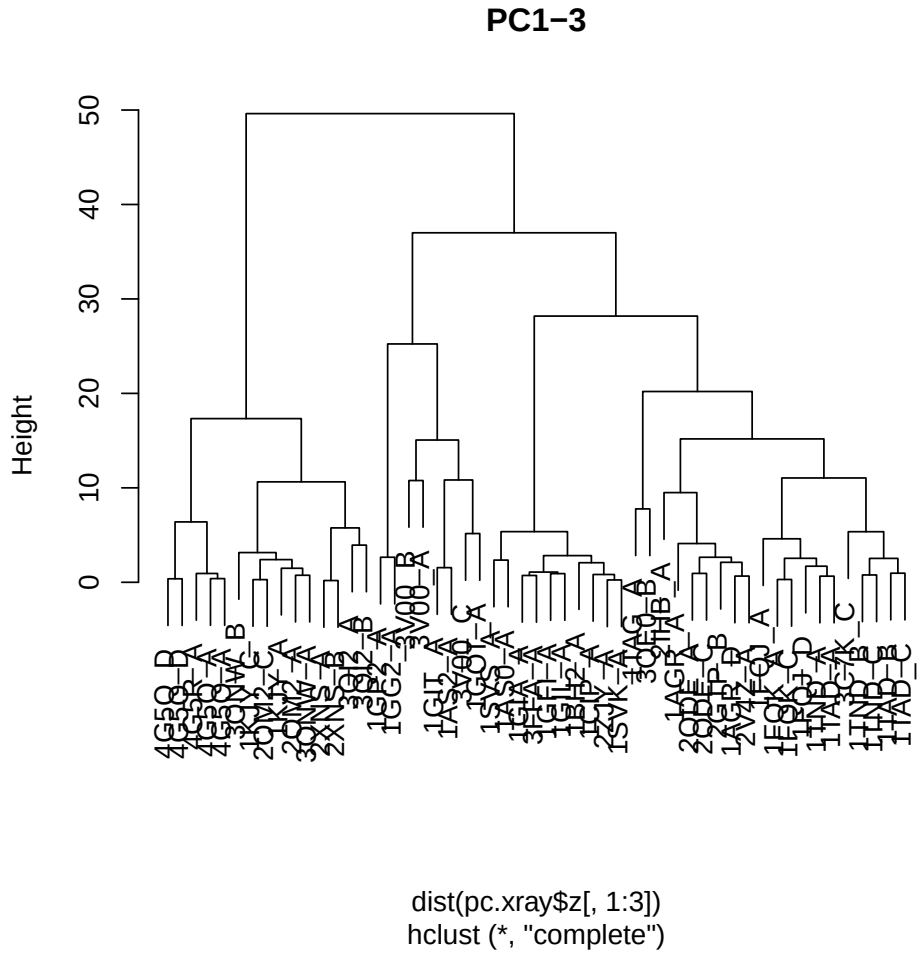
Structure Interpolation: To further aid interpretation, a PDB format trajectory can be produced that interpolates between the most dissimilar structures in the distribution along a given principal cmponent. This involves dividing the difference between the conformers into a number of evenly spaced steps along the principal components, forming the frames of the trajectory. Such trajectories can be directly visualized in a molecular graphics program, such as VMD (Humphrey *et al.*, 1996). Furthermore, the interpolated structures can be analyzed for possible domain and shear movements with the DynDom package (Hayward and Berendsen, 1989), or used as initial seed structures for more advanced reaction path refinement methods such as Conjugate Peak Refinement (Fischer and Karplus, 1992).

```
> 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]) )
```



Conformer Clustering in PC space Clustering of structures in the PC1-3 planes

```
> plot(hclust(dist(pc.xray$z[,1:3])) , labels=pdbs$id, main="PC1-3")
```



2.1.2 Further examples

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.

2.2 Finding available sets of similar structures

First lets examine one way in which we can use bio3d to find sets of similar structures. To collect available transducin crystal structures we first use BLAST to query the PDB to find similar sequences (and hence structures) to our chosen representative (PDB code 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 = 831HJJJ4015
 .
 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.

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)
```

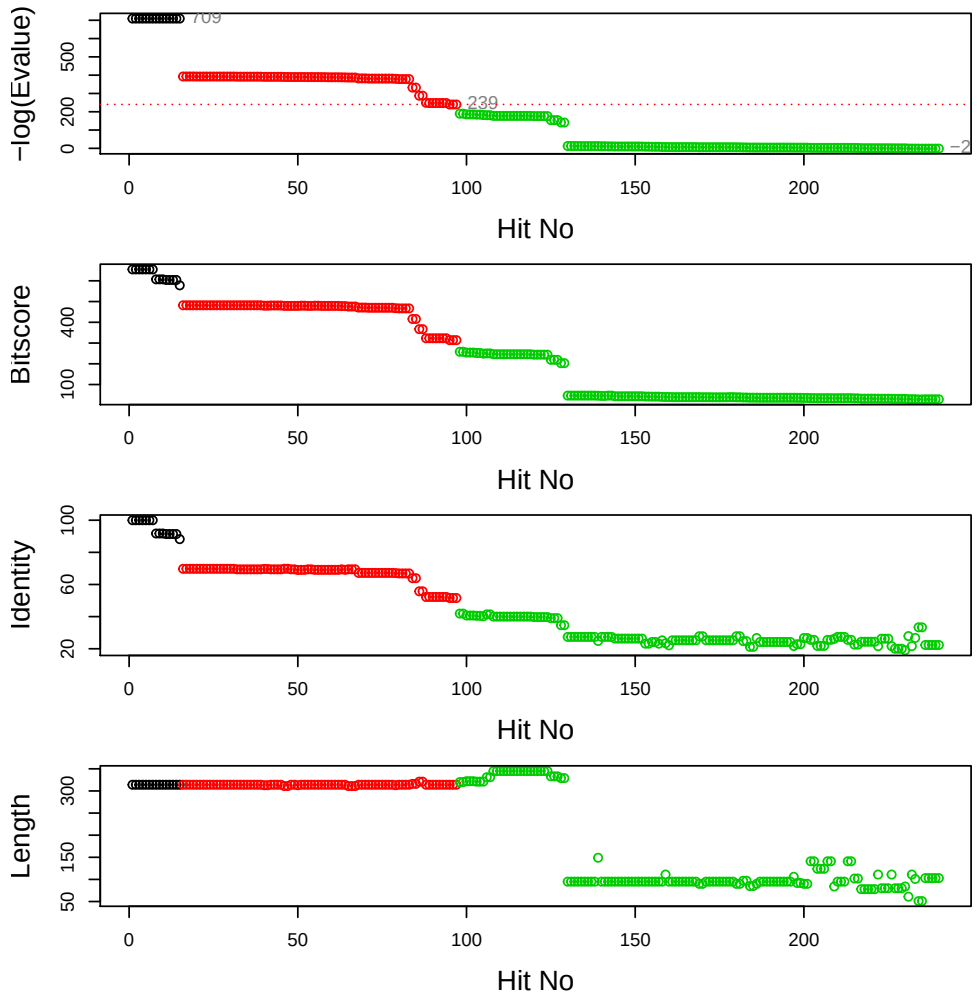
Inspect the alignment with your favorate alignment viewer (e.g. SEAVIEW, available from: http://pbil.univ-lyon1.fr/software/seaview.html). It is possible that you may find some structures with missing residues (or gaps in the alignment) at sites of particular interest (e.g. switch regions in the case of transducin), which you may exclude from the dataset before doing further investigation.

Atom selection: Bio3d provides an atom selection function (`atom.select()`) that returns atom and xyz indicies corresponding to the intersection of a hierarchical selection string. Once an atom selection is made, you can query the properties of the selected atoms, such as their names, residue ids, or coordinates. In a similar fashion, you can set the values of these properties. You can also perform logical operations on atom selections, including finding the intersection or union of two or more atom selections or finding the inverse of the set.

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

```
> rd.raw <- rmsd(pdbs$xyz[,gaps.pos$f.inds])
> rd.fit <- rmsd(pdbs$xyz[,gaps.pos$f.inds], fit=TRUE)
> range(rd.raw)
```

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

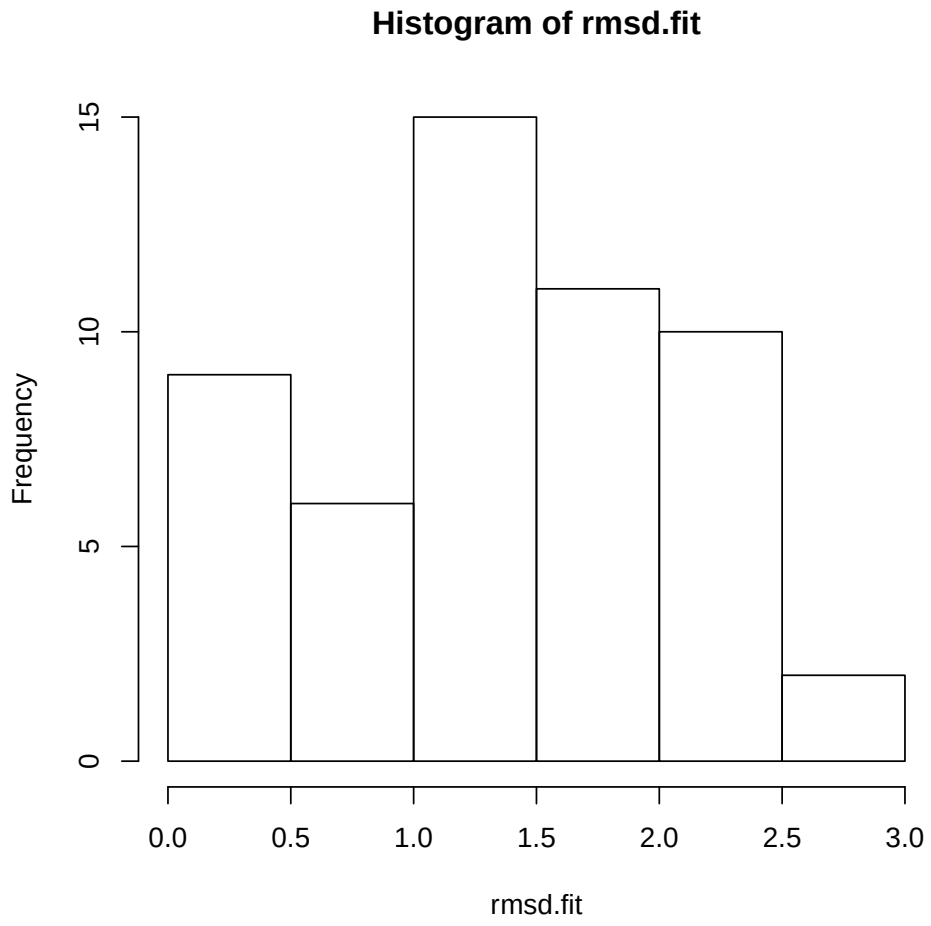


```
[1] 0.000 3.316
```

```
> range(rd.fit)
```

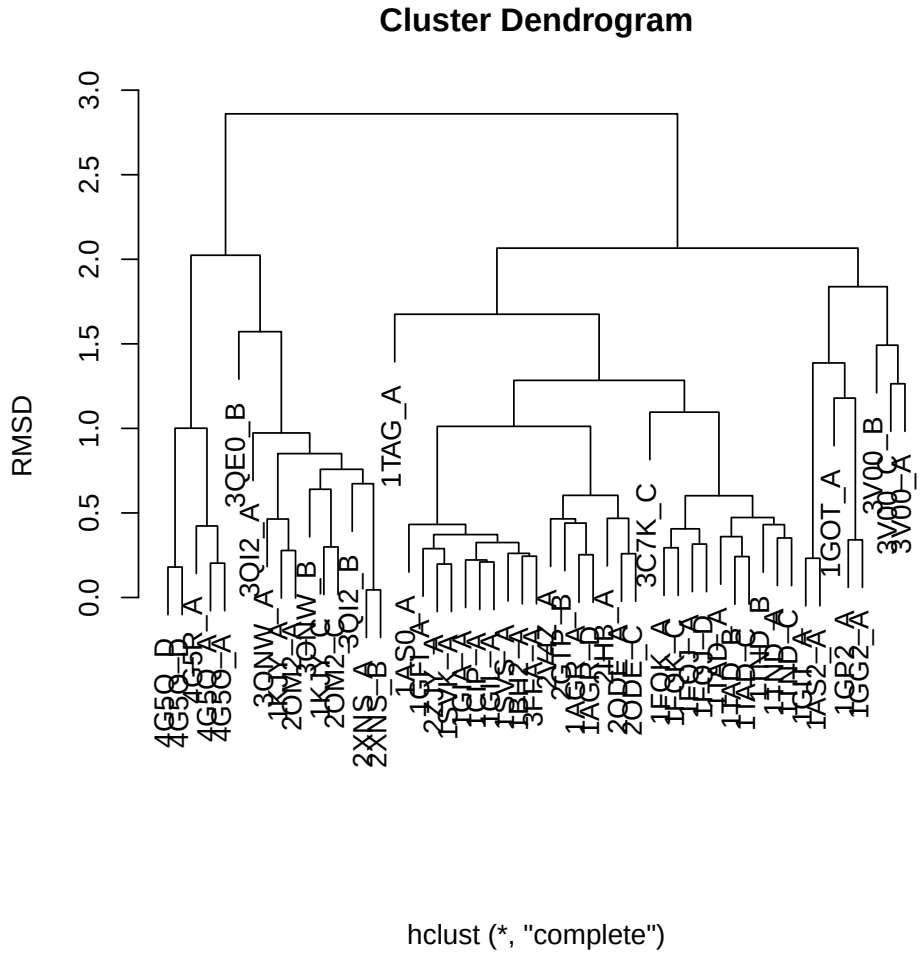
```
[1] 0.00 2.86
```

```
> rmsd.fit <- rmsd(pdbs$xyz[1,gaps.pos$f.inds], pdbs$xyz[,gaps.pos$f.inds], fit=TRUE)
> hist(rmsd.fit)
```



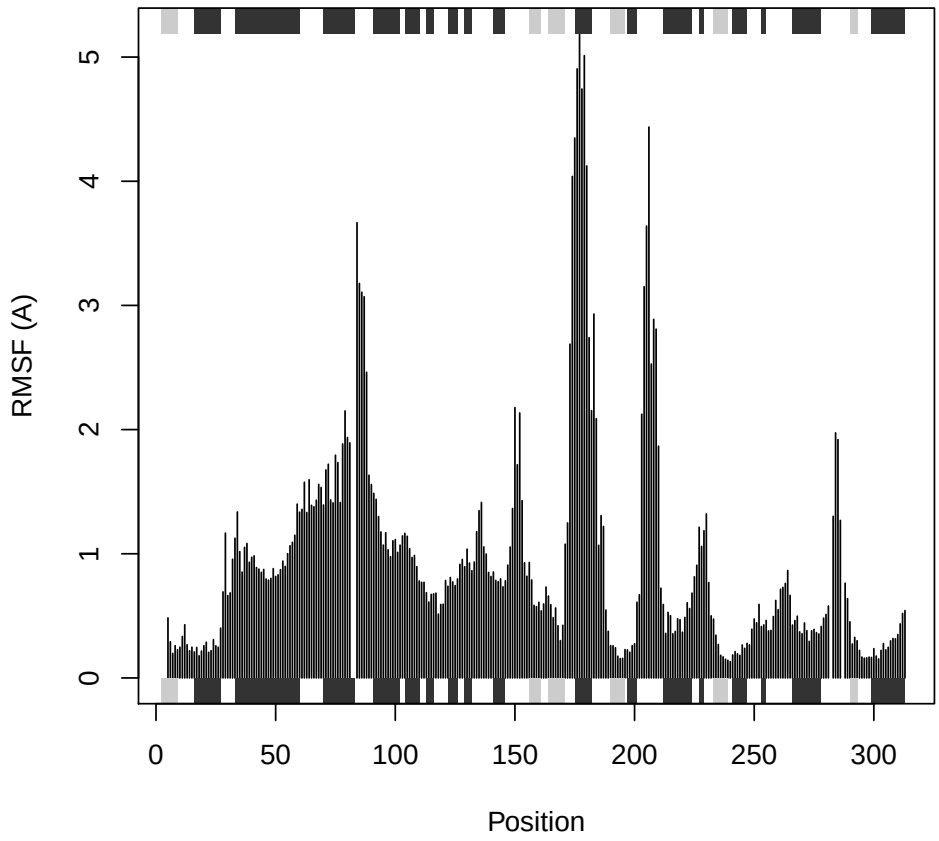
Clustering: Cluster with RMSD or see earler section on PCA [reference back]

```
> plot(hclust(as.dist(rd.fit)), labels=pdbs$id, ylab="RMSD", xlab="")
```



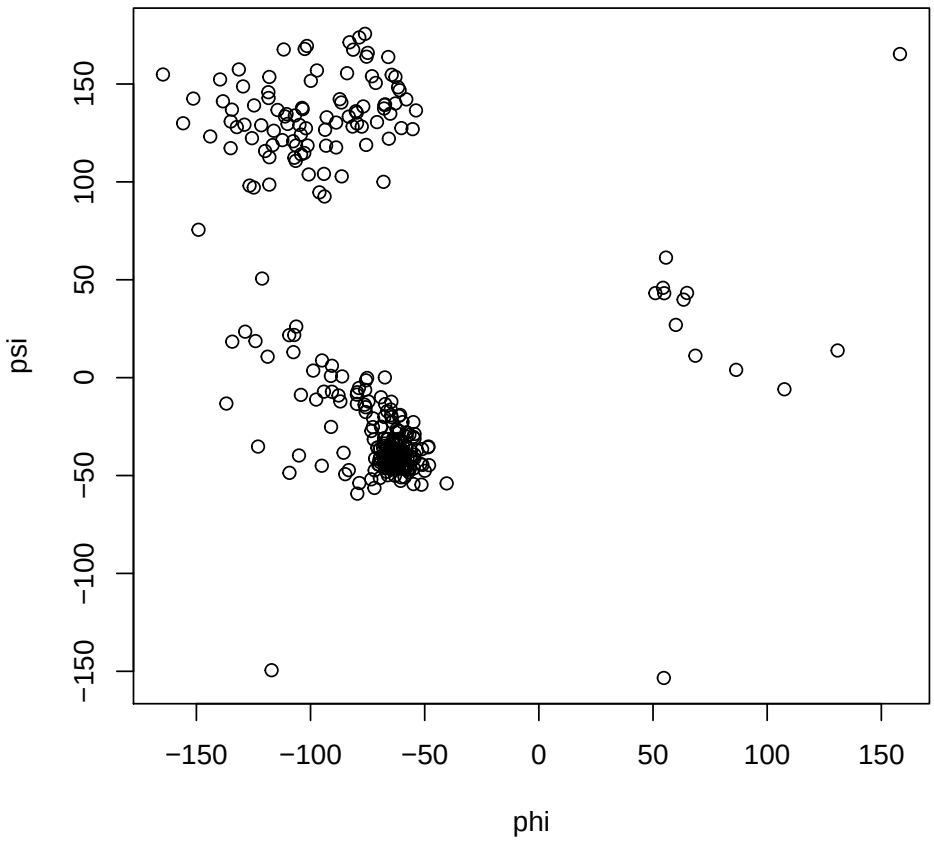
Root mean squared fluctuations (RMSF): RMSF is an often used measure of conformational variance.

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

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



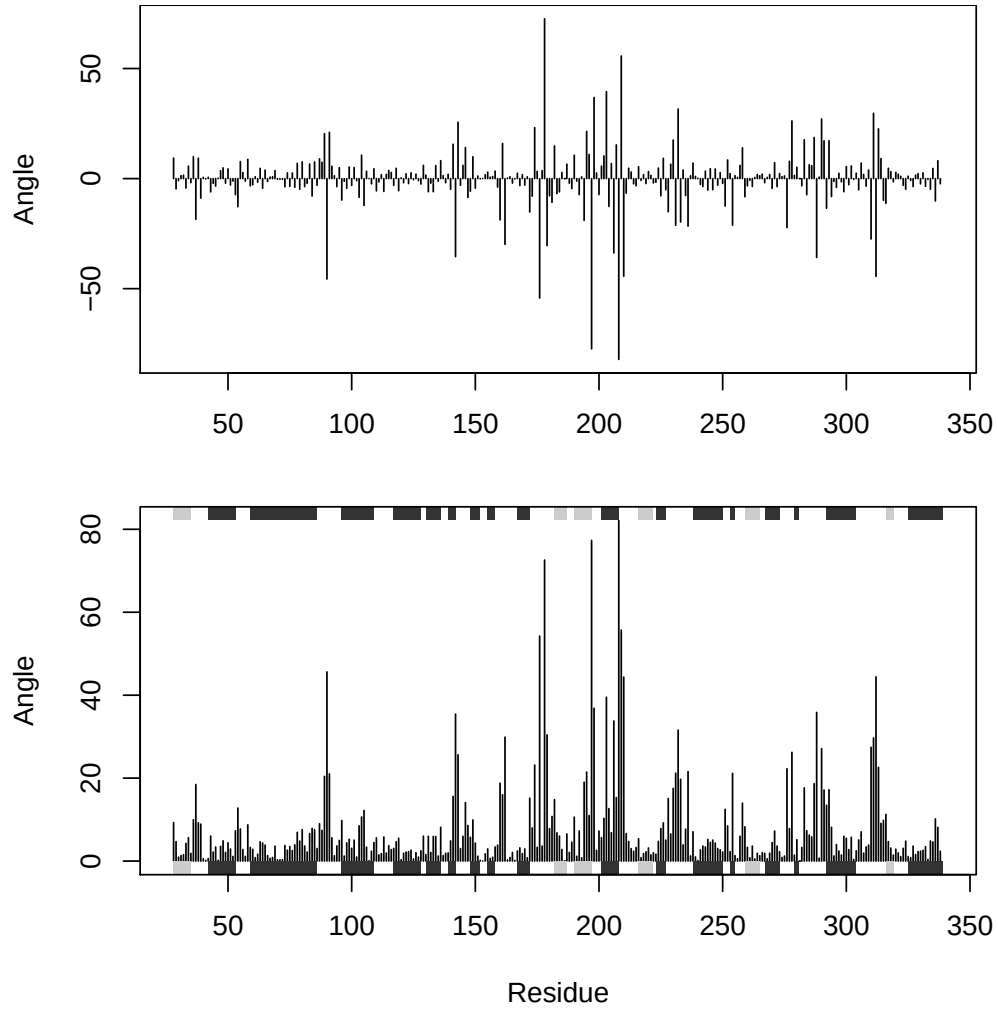
Torsion/Dihedral analysis and Difference distance matrix analysis (DDM): The conformation of a polypeptide or nucleotide chain can be usefully described in terms of angles of internal rotation around its constituent bonds.

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

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

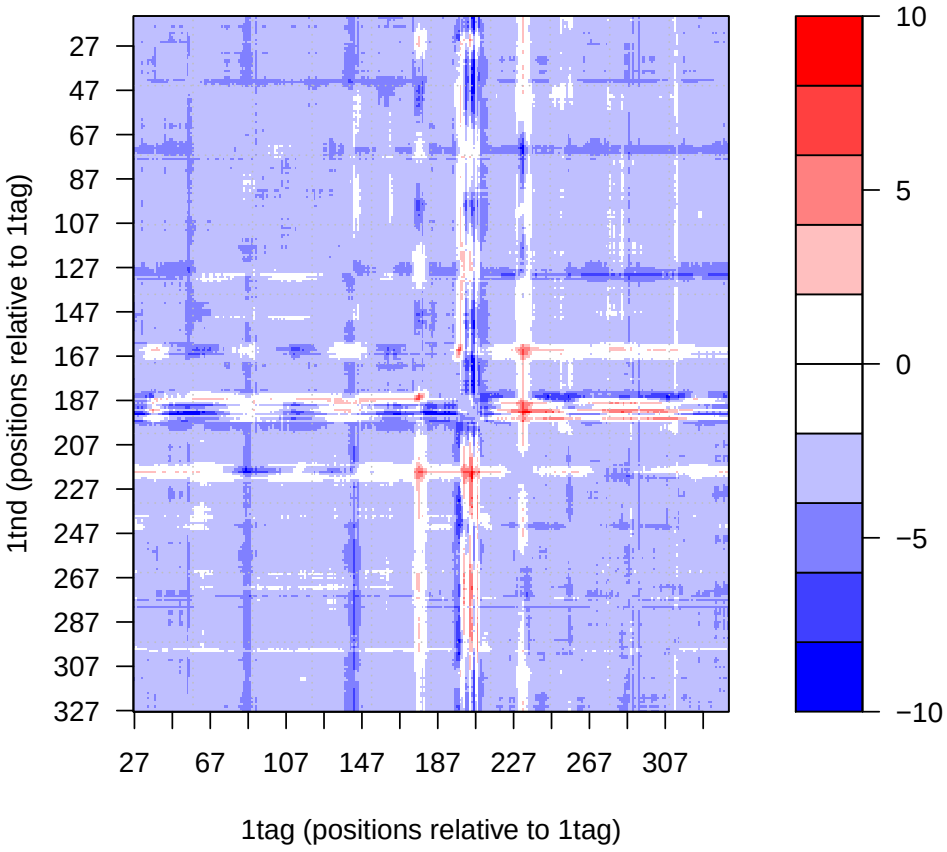



```
> a.xyz <- pdbs$xyz["1TAG_A",]
> b.xyz <- pdbs$xyz["1TND_B",]
> gaps.xyz <- is.gap(pdbs$xyz["1TAG_A",])
> gaps.res <- is.gap(pdbs$ali["1TAG_A",])
> resno <- pdbs$resno["1TAG_A",!gaps.res]
> a <- torsion.xyz(a.xyz[!gaps.xyz],atm.inc=1)
> b <- torsion.xyz(b.xyz[!gaps.xyz],atm.inc=1)
> d.ab <- wrap.tor(a-b)
> #par(mfrow=c(3,1))
> #plot(resno, d.ab, typ="h")
> #plot.bio3d(resno, abs(d.ab), typ="h", sse=sse)
>
> a <- dm(a.xyz[!gaps.xyz])

input is raw 'xyz' thus 'selection' ignored

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

input is raw 'xyz' thus 'selection' ignored

> ddm <- a - b
>
> #plot(ddm, nlevels=10, grid.col="gray", resnum.1=resno, resnum.2=resno,
```

```
> #     xlab="1i6i (positions relative to 1bg2)", ylab="1i5s (positions relative to 1bg2)")
>
>

> 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 (positions relative to 1tag)", ylab="1tnd (positions relative to 1tag)")
```



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

2.3.1 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"))
```

2.3.2 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.

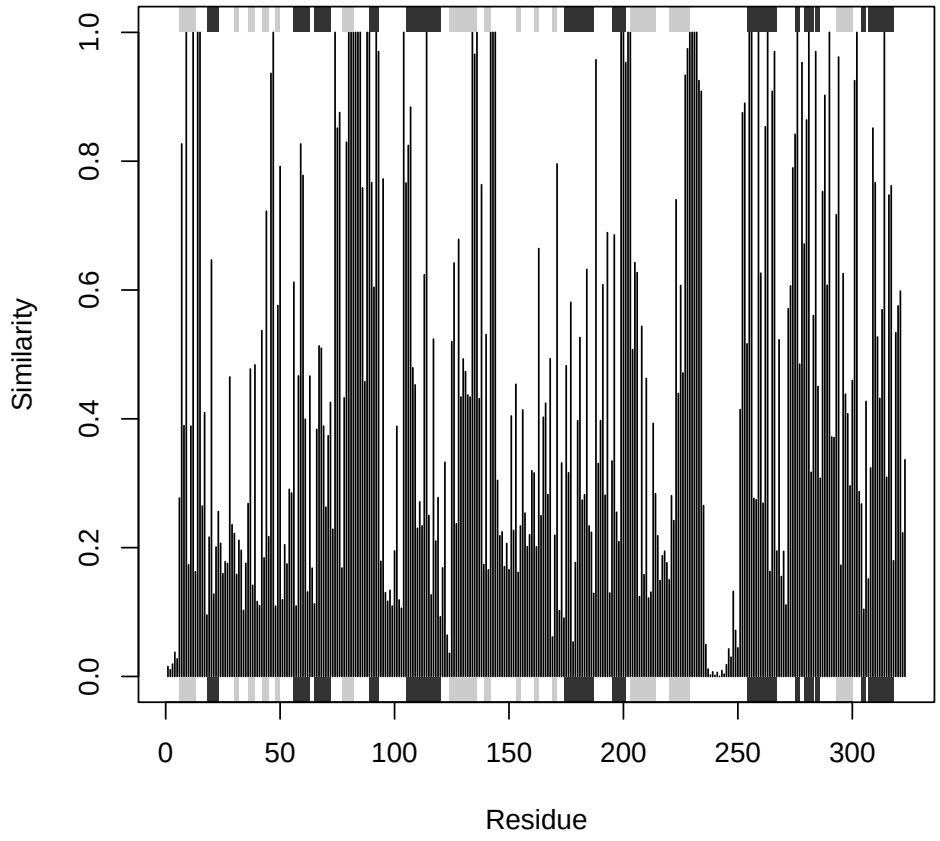
The *similarity* is defined as the average of the similarity scores of all pairwise residue comparisons for that position in the alignment, where the similarity score between any two residues is the score value between those residues in the chosen substitution matrix.

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

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")

> 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")
```




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

Identity, Clustering, Consensus, etc.

Pairwise identity analysis dm plot, histogram, ide.filter

Determine the consensus sequence for a given alignment at a given identity cutoff value.

```
> con <- consensus(aln$ali, cutoff = 0.6)
```

Quantifies residue conservation in a given protein sequence alignment by calculating the degree of amino acid variability in each column of the alignment.

```
> con.sim <- conserv(x=aln$ali, method="similarity", sub.matrix="bio3d")
> ##con.ent <- conserv(x=aln$ali, method="entropy10")
```

plot conservation

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|LC_C
  \item Base packages: base, datasets, graphics, grDevices, methods,
    stats, utils
  \item Other packages: bio3d~2.0
  \item Loaded via a namespace (and not attached): tools~3.0.1
\end{itemize}
```

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