

# Package ‘veriNA3d’

June 19, 2019

**Type** Package

**Title** Biological Structure Analysis

**Version** 0.99.0

**Date** 2018-12-12

**Author** Diego Gallego

**Maintainer** Diego Gallego <diego.gallego@irbbarcelona.org>

**Description** VeriNA3d is an R package for the analysis of Nucleic Acid structural data.

The software was developed on top of bio3d (Grant et al, 2006) with a higher level of abstraction. In addition of single-structure analyses, veriNA3d also implements pipelines to handle whole datasets of mmCIF/PDB structures. As far as we know, no similar software has been previously distributed, thus it aims to fill a gap in the data mining pipelines of PDB structural data analyses.

**License** GPL-3

**Depends** R (>= 3.4)

**Imports** methods, graphics, grDevices, stats, utils, bio3d, circlize, parallel, jsonlite, plot3D, MASS, RColorBrewer, RANN, Rcpp

**LinkingTo** Rcpp

**LazyData** true

**RoxygenNote** 6.1.1

**Roxygen** list(markdown = TRUE)

**Encoding** UTF-8

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** yes

## R topics documented:

applyToPDB . . . . .	2
checkNuc . . . . .	3
CIF-class . . . . .	4

cifAsPDB . . . . .	5
cifDownload . . . . .	6
cifParser . . . . .	7
cif_accessors . . . . .	8
classifyNA . . . . .	10
cleanByPucker . . . . .	12
countEntities . . . . .	13
dataset_wadley2007 . . . . .	13
dssr . . . . .	14
entities . . . . .	15
eRMSD . . . . .	16
fastquery . . . . .	17
findBindingSite . . . . .	18
findHDR . . . . .	19
getAltRepres . . . . .	20
getRNAList . . . . .	21
hasHetAtm . . . . .	22
measureElenoDist . . . . .	22
measureEntityDist . . . . .	24
measureNuc . . . . .	25
pipeNucData . . . . .	26
pipeProtNucData . . . . .	28
plot2D . . . . .	29
plotCategorical . . . . .	30
plotCircularDistribution . . . . .	31
queryAPI . . . . .	32
queryEntryList . . . . .	33
queryFunctions . . . . .	34
queryObsoleteList . . . . .	36
represAsDataFrame . . . . .	36
RMSD . . . . .	37
rVector . . . . .	38
selectModel . . . . .	39
trimByID . . . . .	40
trimSphere . . . . .	41
veriNA3d . . . . .	43
<b>Index</b>	<b>44</b>

---

applyToPDB

*Applies a function over a list of PDB ID entries.*

---

## Description

Given a function of interest, it is applied to all the PDB entries. See supported functions in ?query-Functions.

**Usage**

```
applyToPDB(listpdb = NULL, FUNCTION, as.df = TRUE, cores = 1,  
  progressbar = TRUE, ...)
```

**Arguments**

listpdb	A list/vector containing the PDB IDs of interest. If NULL, the complete list of PDB entries is downloaded and used.
FUNCTION	A function of interest.
as.df	A logical that stands for "as.data.frame". If TRUE, the output will be returned in the form of a data.frame, otherwise a list.
cores	Number of CPU cores to be used.
progressbar	A logical to print in screen a progress bar.
...	optional arguments to FUNCTION.

**Value**

A data.frame with the PDB IDs (first column) and the output of the function of interest (second column) or a list with the results.

**Author(s)**

Diego Gallego

**Examples**

```
listpdb <- c("1s72", "1bau", "1rna")  
applyToPDB(listpdb, queryTechnique)
```

---

checkNuc

*Check nucleotides*

---

**Description**

From a nucleic acid structure (pdb object), it checks the presence of all nucleotide atoms, bond distances, chain breaks and others.

**Usage**

```
checkNuc(pdb, model = 1, chain = "all", id = NULL, refatm = "C4'",  
  force = FALSE)
```

**Arguments**

pdb	A pdb object as obtained from <code>cifAsPDB</code> or <code>read.cif/read.pdb</code> (bio3d package).
model	A string with the desired model number.
chain	A string with the desired chain id.
id	A string with the ID of the input pdb structure.
refatm	A string with the atom to use to identify the nucleotides. Important to analyse models with just (in example) phosphate atoms, in which <code>refatm</code> should be set to "P" (it was thought when analysing the structure with PDB code: 1Y1Y).
force	A logical to force the analysis. Useful when the function does not recognise a nucleic acid in the structure (e.g. because all bases are non-canonical: 1PBL, 1XV6, 1DV4, etc).

**Value**

A data.frame with the data for every nucleotide.

**Author(s)**

Diego Gallego

**Examples**

```
data <- checkNuc(cifAsPDB("1am0"))
broken <- which(data$Break == TRUE)
data[broken, ] ## See the places in which the chain is broken
```

---

CIF-class

*An S4 class to represent a structure parsed from its mmCIF file.*

---

**Description**

All mmCIF files in the PDB at date 2018-Feb-19th contain (just) 14 common attributes, which are represented in the CIF objects herein with the same names as found in mmCIF documentation <http://mmcif.wwpdb.org/>.

**Slots**

entry The ID code  
 audit\_conform 'mmCIF' dictionary version  
 database\_2 Cross-reference ID codes to other databases  
 pdbx\_database\_status Deposition data  
 audit\_author Authors  
 entity Entities (molecules & ions) in the structure

chem\_comp Residues (ATOM & HETATM) in the structure  
 exptl Experimental technique  
 struct Author description of the structure  
 struct\_keywords Author description key words  
 struct\_asym Chain-entity equivalences  
 atom\_sites Details about the crystallographic cell  
 atom\_type Details about the atoms in structure  
 atom\_site The atomic coordinates

### Author(s)

Diego Gallego

### See Also

To create CIF objects use [cifParser\(\)](#)

---

cifAsPDB

*Coerce CIF S4 object to pdb S3 object as found in bio3d package*

---

### Description

Coerces CIF to pdb class.

### Usage

```
cifAsPDB(cif, model = NULL, chain = NULL, alt = c("A"),
         verbose = FALSE)
```

```
## S4 method for signature 'CIF'
cifAsPDB(cif, model = NULL, chain = NULL,
         alt = c("A"), verbose = FALSE)
```

```
## S4 method for signature 'character'
cifAsPDB(cif, model = NULL, chain = NULL,
         alt = c("A"), verbose = FALSE)
```

### Arguments

cif	A CIF object as obtained from cifParser. It can also accept a 4-character PDB ID.
model	A string with the model number (in case you are only interested in a particular model) If NULL, all models are parsed and can be selected afterwards using selectModel. By default, the "atom" attribute of the output will contain only the first model.

chain	A string with the chain identifier (in case you are only interested in a particular chain). If NULL, all chains are included.
alt	A string or a vector of strings with the desired alternative records.
verbose	A logical indicating whether to print details of the process.

**Value**

A pdb object compatible with bio3d (Grant et al. 2006) functions.

**Author(s)**

Diego Gallego

**Examples**

```
cif <- cifParser("1bau")
pdb <- cifAsPDB(cif)
```

---

cifDownload

*Download Protein Data Bank structures*

---

**Description**

Given a 4-character string (PDB ID), download structure.

**Usage**

```
cifDownload(pdbID, path = "default", destfile = NULL,
  extension = ".cif.gz", URL = NULL, verbose = FALSE)
```

**Arguments**

pdbID	A 4 character string that matches a structure in the Protein Data Bank.
path	The path to save the downloaded file. In basic use of the function, the "default" path is a temporal directory, which is removed after the R session is terminated. Some users might prefer to save permanently the files in a local path. This advanced use of the function can be acquired creating (manually) a local hidden directory called ".veriNA3d_mmCIF_files" with <code>dir.create("~/veriNA3d_mmCIF_files")</code> . This function will automatically recognize the new "default" path and use it from there on. Note that the function will never overwrite existing files.
destfile	File name to save the downloaded structure. If NULL, the file name is constructed based on the PDB ID and extension.
extension	A string with the desired file extension.
URL	A string with the URL of use. If NULL, the RCSB is used as default.
verbose	A logical to print process details.

**Value**

The file name.

**Author(s)**

Diego Gallego

**Examples**

```
## cifDownload("1bau")
```

---

cifParser

*Parse coordinates from CIF files*

---

**Description**

Given a file or PDB ID, the function parses the coordinates of the structure. It can also read all the fields of the mmCIF format.

**Usage**

```
cifParser(pdbID, verbose = FALSE, cache = TRUE)
```

```
## S4 method for signature 'ANY'
```

```
cifParser(pdbID, verbose = FALSE, cache = TRUE)
```

**Arguments**

pdbID	A 4-character string that matches a structure in the Protein Data Bank (or an existing file in disk).
verbose	A logical indicating whether to print details of the process.
cache	A logical indicating whether to save mmCIF in RAM. Only one structure will be saved a time.

**Value**

A S4 CIF object.

**Author(s)**

Diego Gallego

**Examples**

```
cif <- cifParser("1bau")
```

---

`cif_accessors`*Accessors to a CIF object*

---

**Description**

S4 method to access the contents of CIF objects.

**Usage**`cifEntry(x)``cifAudit_conform(x)``cifDatabase_2(x)``cifPdbx_database_status(x)``cifAudit_author(x)``cifEntity(x)``cifChem_comp(x)``cifExptl(x)``cifStruct(x)``cifStruct_keywords(x)``cifStruct_asym(x)``cifAtom_sites(x)``cifAtom_type(x)``cifAtom_site(x)`

```
## S4 method for signature 'CIF'
cifEntry(x)
```

```
## S4 method for signature 'CIF'
cifAudit_conform(x)
```

```
## S4 method for signature 'CIF'
cifDatabase_2(x)
```

```
## S4 method for signature 'CIF'
```



```
cifPdbx_database_status(x)

## S4 method for signature 'CIF'
cifAudit_author(x)

## S4 method for signature 'CIF'
cifEntity(x)

## S4 method for signature 'CIF'
cifChem_comp(x)

## S4 method for signature 'CIF'
cifExptl(x)

## S4 method for signature 'CIF'
cifStruct(x)

## S4 method for signature 'CIF'
cifStruct_keywords(x)

## S4 method for signature 'CIF'
cifStruct_asym(x)

## S4 method for signature 'CIF'
cifAtom_sites(x)

## S4 method for signature 'CIF'
cifAtom_type(x)

## S4 method for signature 'CIF'
cifAtom_site(x)
```

### Arguments

x                      a CIF object.

### Value

```
* {cifEntry} `Character` with the mmCIF PDB ID.
* {cifAudit_conform} `Character` vector with dictionary version.
* {cifDatabase_2} `Data.frame` with cross-references.
* {cifPdbx_database_status} `Character` vector with deposition data.
* {cifAudit_author} `Data.frame` with author names.
* {cifEntity} `Data.frame` with molecules & ions in the structure.
* {cifChem_comp} `Data.frame` with residues records in the structure.
* {cifExptl} `Character` vector with experimental technique.
* {cifStruct} `Character` vector with author description of the
  structure.
* {cifStruct_keywords} `Character` vector with author selected key
```

```

    words.
* {cifStruct_asym} `Data.frame` with chain-entity equivalences.
* {cifAtom_sites} `Character` vector with details about the
  crystallographic cell.
* {cifAtom_type} `Data.frame` with about the atoms in structure.
* {cifAtom_site} `Data.frame` with atomic coordinates.

```

### Author(s)

Diego Gallego

### References

Official site of mmCIF dictionary specifications <http://mmcif.wwpdb.org/>

### Examples

```

cif <- cifParser("1bau")
coordinates <- cifAtom_site(cif)

```

---

classifyNA

*Classify RNA or DNA structures*

---

### Description

classifyRNA classifies a structure in the following groups:

- NoRNA: the structure does not contain RNA or it is shorter than a threshold set by "length".
- nakedRNA: the only molecule(s) in the structure is RNA. It can include ionic ligands, but not organic ligands.
- protRNA: the PDB contains a protein-RNA complex.
- DprotRNA: the PDB contains a protein-RNA complex and the protein has D aminoacids.
- DNARNA: the PDB contains DNA-RNA.
- PNARNA: the PDB contains PNA-RNA.
- ANARNA: the PDB contains ANA-RNA.
- LNARNA: the PDB contains LNA-RNA.

- **ligandRNA**: the RNA is interacting with an organic ligand. Ions are not considered as ligands in this class.

### Usage

```
classifyRNA(pdbID, length = 3, force = FALSE, ...)
```

```
classifyDNA(pdbID, force = FALSE, ...)
```

### Arguments

<code>pdbID</code>	A 4-character string that matches a structure ID in the Protein Data Bank.
<code>length</code>	A positive integer. Minimum number of nucleotides to consider RNA as a polymer. An RNA shorter than this threshold is classified in the NoRNA group.
<code>force</code>	A logical to force the query instead of getting presaved data.
<code>...</code>	Arguments to be passed to query function (see <code>?queryFunctions</code> ).

### Details

`classifyDNA` classifies a structure in the following groups:

- **NoDNA**: the structure does not contain DNA.
- **nakedDNA**: the only molecule(s) in the structure is DNA. It can include ionic ligands, but not organic ligands.
- **protDNA**: the PDB contains a protein-DNA complex.
- **DprotDNA**: the PDB contains a protein-DNA complex and the protein has D aminoacids.
- **DNARNA**: the PDB contains DNA-RNA.
- **PNADNA**: the PDB contains PNA-DNA.
- **ligandDNA**: the DNA is interacting with an organic ligand. Ions are not considered as ligands in this class.

In `classifyRNA`, nucleic acid hybrids are considered RNA, while in `classifyDNA` they are considered DNA (e.g. `pdb ID 2HVR`).

### Value

A string with the type of RNA.

**Author(s)**

Diego Gallego

**Examples**

```
classifyRNA("1S72")
```

---

cleanByPucker

---

*Subset nucleotide data according with puckering*


---

**Description**

Function to clean raw data after the pipeline [pipeNucData\(\)](#). It takes a data.frame that should have the columns "pu\_phase", "delta" and "Dp", and returns the nucleotides matching the desired puckering state.

**Usage**

```
cleanByPucker(ntinfo, surenorth = FALSE, suresouth = FALSE,
  pucker = "C3'endo", range = NULL, verbose = TRUE)
```

**Arguments**

ntinfo	A data.frame. The output of <a href="#">pipeNucData()</a> .
surenorth	A logical to return nucleotides in north with restrictions in delta and Dp.
suresouth	A logical to return nucleotides in south with restrictions in delta and Dp.
pucker	A string with the puckering state of interest. Only necessary if surenorth and suresouth are FALSE and range is NULL. When using this option, only the phase is used to subset the data.
range	A numeric vector with the desired phaserange. Only used if no other argument above could be applied.
verbose	A logical to print details of the process.

**Value**

An integer vector with the nucleotides (ntID) matching the desired puckering state.

**Author(s)**

Diego Gallego

**Examples**

```
ntinfo <- pipeNucData("1bau")
north_ntID <- cleanByPucker(ntinfo, surenorth=TRUE)
north <- ntinfo[ntinfo$ntID %in% north_ntID,]
```

---

countEntities	<i>Count entities</i>
---------------	-----------------------

---

**Description**

For a given pdbID, the function gets the Entity data and counts the number of instances of the different entities (e.g. the number of different RNA, the number of different proteins...).

**Usage**

```
countEntities(pdbID, force = FALSE, ...)
```

**Arguments**

pdbID	A 4-character string that matches a structure ID in the Protein Data Bank.
force	A logical to force the query instead of getting presaved data.
...	Arguments to be passed to query function (see ?queryFunctions).

**Value**

A list with the number of instances of each entity.

**Author(s)**

Diego Gallego

**Examples**

```
countEntities("1S72")
```

---

dataset_wadley2007	<i>Dataset of RNA structures by Wadley et al.</i>
--------------------	---

---

**Description**

Data.frame containing the list of PDB IDs of the structures used by Wadley et al. in their eta-theta study about RNA.

**Usage**

```
data(dataset_wadley2007)
```

**Format**

An object of class `data.frame`

**pdb:** PDB ID.

**chain:** Chain

**Value**

`data.frame` with list of PDB IDs.

**References**

Wadley, L.M., K.S. Keating, C.M Duarte, and A.M. Pyle. 2007.  
 “Evaluating and Learning from Rna Pseudotorsional Space:  
 Quantitative Validation of a Reduced Representation for  
 RNA Structure.” *Journal of Molecular Biology* 372 (4): 94257

---

dssr

*Dissecting the Spatial Structure of RNA with DSSR*

---

**Description**

Wrapper function to execute DSSR (see reference below) on a DNA or RNA structure and parse the result.

**Usage**

```
dssr(pdb, exefile = "x3dna-dssr", dssrargs = c("--nmr", "--torsion360",
  "--more"), verbose = FALSE)
```

**Arguments**

pdb	It can be: <ul style="list-style-type: none"> <li>• A 4 character string corresponding to a PDB ID.</li> <li>• A <code>pdb/mmcif</code> file.</li> <li>• A <code>pdb</code> object as provided by <code>cifAsPDB()</code> or <code>bio3d::read.pdb()</code>.</li> </ul>
exefile	A string with the program name.
dssrargs	A vector of strings with the desired arguments to feed DSSR.
verbose	A logical indicating whether to print details of the process.

**Value**

A list with the json output of DSSR.

**Author(s)**

Diego Gallego

## References

Lu, X.J., H.J. Bussemaker, and W.K. Olson. 2015. "DSSR: An Integrated Software Tool for Dissection the Spatial Structure of RNA." *Nucleic Acids Research* 43 (21): e142

## Examples

```
# Not run
# dssr_1bau <- dssr("1bau")
```

---

entities

*List of NA containing PDB IDs and related data*

---

## Description

Number of each type of entity in a set of Nucleic Acid containing structures.

## Usage

```
data(entities)
```

## Format

An object of class data.frame

**pdbID:** PDB ID.

**RNA:** Number of different RNA entities.

**DNA:** Number of different DNA entities.

**Hybrid:** Number of different Hybrid DNA/RNA entities.

**PNA:** Number of different PNA entities.

**Prot:** Number of different Protein (L) entities.

**Dprot:** Number of different Protein (D) entities.

**Ligands:** Number of different ligands.

**Water:** It's 1 when there's water in the structure and 0 when it is not.

**Other:** Number of different "other" entities.

## Value

data.frame with a list and features of NA PDB IDs.

eRMSD

*Compute the epsilon RMSD between two RNA structures***Description**

Given two RNA with the same length, the functions calculates its epsilon RMSD (Bottaro et al. 2014), reproducing BaRNABA software (Bottaro et al. 2018).

**Usage**

```
eRMSD(cif1 = NULL, cif2 = NULL, pdb1 = NULL, pdb2 = NULL,
      rvector1 = NULL, rvector2 = NULL)
```

```
## S4 method for signature 'CIF,CIF'
eRMSD(cif1 = NULL, cif2 = NULL)
```

```
## S4 method for signature 'ANY,ANY'
eRMSD(pdb1 = NULL, pdb2 = NULL, rvector1 = NULL,
      rvector2 = NULL)
```

**Arguments**

cif1	A CIF object as obtained from <a href="#">cifParser()</a> .
cif2	A CIF object as obtained from <a href="#">cifParser()</a> .
pdb1	A pdb object as obtained from <a href="#">cifAsPDB</a> or <a href="#">read.cif/read.pdb</a> (from <a href="#">bio3d</a> package).
pdb2	A pdb object as obtained from <a href="#">cifAsPDB</a> or <a href="#">read.cif/read.pdb</a> (from <a href="#">bio3d</a> package).
rvector1	A data.frame as obtained from <a href="#">rVector()</a> using <code>simple_out=TRUE</code> .
rvector2	A data.frame as obtained from <a href="#">rVector()</a> using <code>simple_out=TRUE</code> .

**Value**

A numeric with the epsilon RMSD between the two structures.

**Author(s)**

Diego Gallego

**References**

Bottaro, S., F. Di Palma, and G Bussi. 2014. "The Role of Nucleobase Interactions in RNA Structure and Dynamics." *Nucleic Acids Research* 42 (21): 1330614

Bottaro, S., G. Bussi, G. Pinamonti, S. Reiber, W. Boomsma, and K. Lindorff-Larsen. 2018. "Barnaba: Software for Analysis of Nucleic Acids Structures and Trajectories." *RNA*, Epub ahead of print



## Examples

```
cif <- cifParser("2d18")
model1 <- selectModel(cif=cif, model=1)
model3 <- selectModel(cif=cif, model=3)
eRMSD(cif1=model1, cif2=model3)
```

---

fastquery

*List of NA containing PDB IDs and related data*

---

## Description

Presaved data to speed up some queries.

## Usage

```
data(fastquery)
```

## Format

An object of class data.frame with the following fields:

**pdbID:** PDB ID.

**Technique:** Experimental technique.

**Resol:** Resolution. For NMR structures it contains an empty string.

**DNAclass:** Output of classifyDNA function.

**RNAclassOver0:** Output of classifyRNA with length=0 or length=1. If the structure has one or two nucleotides, it is also considered an RNA containing structure.

**RNAclassOver2:** Output of classifyRNA with length=3. RNA molecules shorter than 3 are classified as NoRNA.

## Value

data.frame with a list and features of NA PDB IDs.

---

findBindingSite	<i>Function to get data about the atoms in interacting site.</i>
-----------------	--

---

### Description

For pdb structures with protein-nucleic acid complexes, the function finds the atoms in the interacting site. It allows the user to set as reference the nucleic acid, the protein, or particular desired chains.

### Usage

```
findBindingSite(pdb, cutoff = 5, select = "Nuc", nchain = NULL,
  pchain = NULL, hydrogens = FALSE, byres = FALSE, verbose = FALSE,
  ...)
```

### Arguments

pdb	A cif/pdb object obtained from cifParser/read.pdb respectively or a pdb ID so that the function can download the data.
cutoff	A numeric to set the maximum distance for atoms to be returned.
select	A string that should match "Nuc", "Prot", "DNA" or "RNA", to be used as reference.
nchain	A string with the nucleic acid chain to get data about. If NULL, all of them are selected (according with select argument).
pchain	A string with the protein chain to get data about. If NULL, all of them are selected.
hydrogens	A logical to use the hydrogens in the structure or remove them.
byres	A logical to indicate if the output should be referred to the residues rather than atoms.
verbose	A logical to print details of the process.
...	Arguments to selectModel and/or alt records.

### Value

A data.frame with the atomic distances in the interacting site.

### Author(s)

Diego Gallego

### Examples

```
pdb <- cifParser("1b3t") # A protein-DNA complex
data <- findBindingSite(pdb, select="DNA", byres=TRUE)
```

---

findHDR*Find 2D High Density Regions and return list of nucleotides*

---

## Description

Given a data.frame ("ntinfo") the function computes a 2D Kernel Density Estimation between the desired fields and returns a list of nucleotides (according with the ntID column) that are found in High Density Regions of the 2D map.

## Usage

```
findHDR(ntID = NULL, ntinfo, x = "eta", y = "theta", SD_DENS = 1,  
        bandwidths = c(40, 40), dens = NULL, lims = c(0, 360, 0, 360))
```

## Arguments

ntID	an object of class integer with the desired nucleotides of analysis. If NULL all the nucleotides in the data.frame will be used.
ntinfo	a data.frame with the input data. It should contain three columns (additional columns will be ignored). One of them should be "ntID" and the other two are optional and can be specified using the parameters "x" and "y".
x	name of the column that will be used as "x" axis.
y	name of the column that will be used as "y" axis.
SD_DENS	height above the mean to be used to select the nucleotides
bandwidths	object to be passed to the "kde2d" function (only used if "dens" is NULL).
dens	optional object containing the output of "kde2d" or equivalent.
lims	The limits of the rectangle covered by the grid as c(xl, xu, yl, yu).

## Value

a list of vectors containing the nucleotides ID (according with the ntID column) that clusterize in different regions of the 2D diagram.

## Author(s)

Diego Gallego

---

getAltRepres	<i>Get Alternative Representants</i>
--------------	--------------------------------------

---

## Description

This function is closely related with getRNAList(). From its output, the family members of each equivalence class are checked for desired features. The first member that matches all the desired features is returned.

## Usage

```
getAltRepres(rnalist, technique = NULL, resol = NULL, type = NULL,
  length = 3, progressbar = TRUE, verbose = FALSE, as.df = FALSE,
  na.rm = TRUE)
```

## Arguments

rnalist	The output of getRNAList.
technique	One or more techniques of interest (For correct use, see example below). For the list of techniques, see "veriNA3d:::allowedtechs".
resol	A positive real number to specify a desired resolution.
type	A string indicating the type of desired RNA, according with the classifyRNA function.
length	To be passed to classifyRNA.
progressbar	A logical to print in screen a progress bar.
verbose	A logical to print details of the process.
as.df	A logical to return the output as a data.frame.
na.rm	A logical to clean missing values out of the output.

## Value

A data.frame with info about all the "Equivalence Classes" and the selected Representants according with the specified conditions.

## Author(s)

Diego Gallego

## Examples

```
#rnalist <- getRNAList(release=3.2, threshold="1.5A")
#alternative <- getAltRepres(rnalist=rnalist,
#                             type="nakedRNA")
```

---

`getRNAList`*Download Representative List of RNA structures*

---

**Description**

According with Leontis & Zirbel (references below), the PDB contains structures that are redundant. Their server provides weekly releases of Representative Sets of RNA structures (also called non-redundant lists). This function access their website and returns the desired list.

**Usage**

```
getRNAList(release = "current", threshold = "all", as.df = FALSE)
```

**Arguments**

<code>release</code>	A number indicating the list of interest.
<code>threshold</code>	A string that matches one of the lists in the BGSU RNA site ("1.5A", "2.0A", "2.5A", "3.0A", "3.5A", "4.0A", "20.0A", "all"). Note that "all" returns structures solved by any technique.
<code>as.df</code>	A logical to return the output as a data.frame

**Value**

A data frame with the list of Equivalence Classes and the Representant and Members of each Equivalence Class. Note that the output is formatted according with Leontis&Zirbel nomenclature (AAAAIMIC), where "AAAA" is the PDB accession code, "M" is the model and "C" is the Chain to be used.

**Author(s)**

Diego Gallego

**References**

Official site: <http://rna.bgsu.edu/rna3dhub/nrlist/>  
Publication: Leontis, N.B., and C.L. Zirbel. 2012. "Nonredundant 3D Structure Datasets for RNA Knowledge Extraction and Benchmarking."  
In RNA 3D Structure Analysis and Prediction, edited by N. Leontis and E. Westhof, 27:28198. Springer Berlin Heidelberg

**Examples**

```
data <- getRNAList(release=3.2, threshold="1.5A")
```

---

hasHetAtm

*Check if a given PDB contains the ligand/modbase of interest*


---

### Description

Given a 4-character string (PDB ID) and a ligand/modbase ID, the function checks the presence of the ligand/modres in the given PDB ID. To check for the presence of sodium ions use hetAtms="Na" instead of NA. If you are interested on the whole list of heterogeneous atoms see [queryHetAtms\(\)](#).

### Usage

```
hasHetAtm(pdbID, hetAtms)
```

### Arguments

pdbID	A 4-character string.
hetAtms	A string with the ligand/modbase ID.

### Value

A logical. TRUE if the given hetAtms is present in the structure.

### Author(s)

Diego Gallego

### Examples

```
hasHetAtm("1s72", "MG") # Check if structure has Magnesium ion
```

---

measureElenoDist

*Computes distances between the atoms of interest in a mmCIF structure*


---

### Description

Given a pdb object (or a pdb ID), the function computes the distances between the desired atoms and returns the closest ones. Note that eleno numbers might be different in the PDB vs mmCIF formats and this may lead to errors.

### Usage

```
measureElenoDist(pdb, model = NULL, refeleno, eleno, n = 1,
  cutoff = c(0, 5), verbose = FALSE, detailedoutput = TRUE,
  data_of_interest = NULL)
```

**Arguments**

pdb	A pdb object obtained as from <code>cifAsPDB()</code> or <code>read.pdb/read.cif</code> (bio3d functions).
model	The model of interest to use in the calculations. The first model is always the default.
refeleno	A vector of eleno (element number) to take as reference.
eleno	A vector of eleno to measure the distances.
n	An integer indicating how many closests atoms to return. The default n=1 returns only the closest atom; n=2 would return the two closest atoms and so on. If NULL, any number of atoms within the cutoff will be returned.
cutoff	A numeric vector indicating the distance range to consider in angstroms. Atoms further than the cutoff won't be returned.
verbose	A logical indicating whether to print details of the process.
detailedoutput	A logical indicating whether to include additional information for each atom (see <code>data_of_interest</code> below). If FALSE, only the eleno (element number) and distances are returned.
data_of_interest	A vector of strings. Only used if <code>detailedoutput</code> is TRUE. The vector should only contain the strings between the following: "type", "eley", "alt", "resid", "chain", "resno", "insert", "x", "y", "z", "o", "b", "entid", "elesy", "charge", "asym_id", "seq_id", "comp_id", "atom_id", "model". The selected fields will be returned for both atoms.

**Value**

A data.frame with the nearest atom neighbours information. Fields suffixed with '\_A' refer to the atoms used as reference. Fields suffixed with '\_B' refer to the 'contacting'/closest atoms.

**Author(s)**

Diego Gallego

**Examples**

```
## Dowload cif file and save coordinates data
cif <- cifParser("1enn")
coordinates <- cifAtom_site(cif)

## Find atom numbers for desired entities (e.g. water and DNA)
water_eleno <- coordinates[coordinates$label_atom_id == "O", "id"]
dna_eleno <- coordinates[coordinates$label_comp_id %in%
  c("DA", "DT", "DG", "DU"), "id"]

## Find which DNA atoms are in 5 Angstroms distance from the water
data <- measureElenoDist(cif, refeleno=water_eleno, eleno=dna_eleno,
  n=NULL, cutoff=5)

## To see the data
```

```
head(data)
```

---

measureEntityDist	<i>Computes distances between all the atoms of selected entities in a mm-CIF structure</i>
-------------------	--

---

## Description

Given a cif object (or a pdb ID), the function computes the distances between atoms of the selected entity IDs. For each atom/residue of the reference entity the function returns the closest atoms of the other entities. This function is a wrapper of [measureElenoDist\(\)](#) and simplifies its use. If you are unfamiliar with the concept of entities in a mmCIF structure see example below.

## Usage

```
measureEntityDist(cif, model = NULL, refent, entities = c("all"), ...)
```

## Arguments

cif	A cif object obtained from <a href="#">cifParser</a> or a pdb ID.
model	The model of interest to use in the calculations. The first model is always the default.
refent	A string with the entity ID of reference. The distance output will be referred to the atoms/residues of this entity.
entities	A character vector with the entities of interest. The default "all" will select all of them except the refent.
...	Additional arguments to be passed to <a href="#">measureElenoDist()</a> .

## Value

A data.frame with the nearest atoms neighbour information.

## Author(s)

Diego Gallego

## Examples

```
## To see the entities of a given structure use:
cif <- cifParser("1enn")
cifEntity(cif)

## Suppose you are interested on the interactions of water and DNA
water_entity <- 5
dna_entity <- 1

## Find which DNA atoms are in 5 Angstroms distance from the water
```



```

data <- measureEntityDist(cif, refent=water_entity,
  entities=dna_entity, n=10, cutoff=5)
## An equivalent run without downloading the cif file previously
data <- measureEntityDist("1enn", refent=water_entity,
  entities=dna_entity, n=10, cutoff=5)

## This option is better than using the example in ?measureElenoDist,
## since this way it would also take into account modified residues, if
## any.

```

measureNuc

*Obtain desired nucleotide measurements*

## Description

From a nucleic acid structure (pdb object), it computes the desired atomic distances, angles, dihedral angles, puckering conformation and Dp distance (See definition of Dp in MolProbity paper by Chen et al. 2010).

## Usage

```

measureNuc(pdb, model = 1, chain = "all", v_shifted = TRUE,
  b_shifted = TRUE, distances = "default", angles = "default",
  torsionals = "default", pucker = TRUE, Dp = TRUE, refatm = "C4'",
  force = FALSE)

```

## Arguments

pdb	A pdb object as obtained from cifAsPDB or read.cif/read.pdb (bio3d package).
model	A string with the desired model number.
chain	A string with the desired chain id.
v_shifted	A logical. If TRUE, puckering angles (nu0 to nu4) are returned in the range 0 to 360 degrees. Otherwise, -180 to +180.
b_shifted	A logical. If TRUE, backbone angles, chi and kappa are returned in the range 0 to 360 degrees. Otherwise, -180 to +180.
distances	A data.frame indicating all the intra and inter-nucleotide atomic distances of interest. See details section. A default option is preconfigured to simplify the use of the function and can be seen typing 'veriNA3d::distances'.
angles	A data.frame indicating all the intra and inter-nucleotide angles of interest. See details section. A default option is preconfigured to simplify the use of the function and can be seen typing 'veriNA3d::angles'.
torsionals	A data.frame indicating all the intra and inter- nucleotide torsional angles of interest. See details section. A default option is preconfigured to simplify the use of the function and can be seen typing 'veriNA3d::torsionals'.
pucker	A logical indicating whether to compute the puckering.

Dp	A logical indicating whether to compute the Dp distance.
refatm	A string with the atom to use to identify the nucleotides. Important to analyse models with just (in example) phosphate atoms, in which refatm should be set to "P" (it was thought when analysing the structure with PDB code: 1Y1Y).
force	A logical to force the analysis. Useful when the function does not recognise a nucleic acid in the structure (e.g. because all bases are non-canonical: 1PBL, 1XV6, 1DV4, etc).

### Details

The format of 'distances', 'angles' and 'torsionals' is: First column should indicate the first atom, second column second atom (and so on in the case of angles and torsional angles). An extra last column is optional and should contain the names to identify each measurement in the output. Plane atom names are interpreted as intra- nucleotide measurements. For inter-nucleotide measurements use the prefix "pre\_" or "post\_" before the atom name. In example, to compute all inter-phosphate distances, use as argument:

```
distances=data.frame(atomA=c("P"), atomB=c("post_P"), labels=c("interphosphate"), stringsAsFactors=FALSE).
```

### Value

A data.frame with the measurements for every nucleotide.

### Author(s)

Diego Gallego

### Examples

```
distances <- data.frame(atomA=c("P"), atomB=c("post_P"),
                        labels=c("interphosphate"),
                        stringsAsFactors=FALSE)
measureNuc(cifAsPDB("1bna"), distances=distances, angles=NULL,
           torsionals=NULL, Dp=NULL)
```

---

pipeNucData

*Obtain nucleotide details from a data set of RNA structures*

---

### Description

Pipeline to generate a data.frame with the desired info for a list of PDB. Nucleotides are labeled with a unique identifier (column ntID).

**Usage**

```
pipeNucData(pdbID, model = NULL, chain = NULL, range = c(3, 1e+05),
  path = NULL, extension = NULL, cores = 1, progressbar = TRUE,
  ...)
```

**Arguments**

pdbID	A list/vector containing the desired PDB IDs or a list of pdb objects as provided by "read.pdb", "read.cif", "cifParser" ...
model	A vector with same length of pdbID containing the desired model for each pdbID. If all models are desired, use "all". If no models are specified, the first one will be used for each pdbID.
chain	A vector with same length of pdbID containing the desired chain for each pdbID. If no chain is specified, all chains will be analysed by default. Non-nucleic acid chains will be ignored.
range	A numeric vector with two values to indicate the desired length range for the Nucleic Acid chains. If a chain falls outside the range, it is not analysed.
path	Directory in which the PDB/CIF files can be found (if NULL, the function will download them). If you provide a "path", make sure the file names are the PDB IDs followed by ".cif" or ".pdb". The function will find them using the strings in pdbID, so make sure you use the same case.
extension	A string matching the files extension (e.g. ".pdb", ".cif", "pdb.gz", "cif.gz"). Only necessary if the PDB files are to be read from disk and a path is provided.
cores	Number of CPU cores to be used.
progressbar	A logical to print in screen a progress bar.
...	Arguments to be passed to <a href="#">measureNuc()</a> .

**Value**

A data.frame with data about every nucleotide in the input set.

**Author(s)**

Diego Gallego

**Examples**

```
## This is a toy example, see vignettes for real-world usages.
pdblist <- list("1bau", "2rn1")
model <- list("1", "2")
chain <- list("all", "all")
ntinfo <- pipeNucData(pdbID=pdblist, model=model, chain=chain)
```

---

pipeProtNucData	<i>Obtain nucleotide-protein interactions from a data set of structures</i>
-----------------	---

---

## Description

Pipeline to generate a data.frame with the data about the closest nucleotides to the protein for a list of PDB. The data can be related to unique nucleotide identifiers (ntID) by providing the output of the independent pipeline [pipeNucData\(\)](#).

## Usage

```
pipeProtNucData(pdbID, model = NULL, chain = NULL, ntinfo = NULL,
  path = NULL, extension = NULL, cores = 1, progressbar = TRUE,
  cutoff = 15, ...)
```

## Arguments

pdbID	A list/vector containing the desired PDB IDs or a list of pdb objects as provided by "read.pdb", "read.cif", "cifParser" ...
model	A vector with same length of pdbID containing the desired model for each pdbID. If all models are desired, use "all". If no models are specified, the first one will be used for each pdbID.
chain	A vector with same length of pdbID containing the desired chain for each pdbID. If no chain is specified, all chains will be analysed by default. Non-nucleic acid chains will be ignored.
ntinfo	Optional. A data.frame obtained from <a href="#">pipeNucData()</a> for the same dataset (or bigger), but not smaller.
path	Directory in which the PDB/CIF files can be found (if NULL, the function will download them). If you provide a "path", make sure the file names are the PDB IDs followed by ".cif" or ".pdb". The function will find them using the strings in pdbID, so make sure you use the same case.
extension	A string matching the files extension (e.g. ".pdb", ".cif", ".pdb.gz", ".cif.gz"). Only necessary if the PDB files are to be read from disk and a path is provided.
cores	Number of CPU cores to be used.
progressbar	A logical to print in screen a progress bar.
cutoff	A numeric with the maximum distance to return. To be passed to <a href="#">findBindingSite()</a> .
...	Additional arguments to be passed to <a href="#">findBindingSite()</a> .

## Value

A data.frame with data about the atomic distances in the interacting sites of every structure in the input set.

**Author(s)**

Diego Gallego

**Examples**

```
## This is a toy example, see vignettes for more usages.
pdblist <- list("1nyb", "2ms1")
aantinfo <- pipeProtNucData(pdbID=pdblist)
```

plot2D

*Make 2D data plot or 3D view of the density***Description**

Function to make a plot highlighting the most populated regions.

**Usage**

```
plot2D(ntinfo, ntID = NULL, dens = NULL, bandwidths = NULL,
       x = "eta", y = "theta", drawcontour = TRUE,
       sd_over_mean_contours = c(1, 2, 4), etatheta = FALSE,
       points = NULL, colpoints = "red", file = NULL, width = 15,
       height = 15, bg = "white", units = "cm", res = 200)

plot3Ddens(ntinfo, ntID = NULL, dens = NULL, bandwidths = NULL,
           x = "eta", y = "theta", defaultview = NULL, thetaplot, phiplot,
           cleanerview = FALSE, file = NULL, width = 15, height = 15,
           bg = "white", units = "cm", res = 600)
```

**Arguments**

ntinfo	A data.frame with the input data. It should contain the columns with the desired data and a column labeled ntID.
ntID	A vector of integers with the desired nucleotides of analysis. If NULL all the nucleotides in the data.frame will be used.
dens	The output of a kernel density estimation (e.g. kde2d function) over the data of interest. If it is NULL and drawcontour=TRUE, it will be computed on the fly.
bandwidths	In case dens=NULL and drawcontour=TRUE, it will be passed to kde2d().
x	A string with the parameter to be placed in the x axis.
y	A string with the parameter to be placed in the y axis.
drawcontour	A logical to highlight the most populated regions of the plot.
sd_over_mean_contours	A numeric vector with the standard deviations over the mean to plot the contours (in case drawcontour=TRUE).

etatheta	A logical to return a eta-theta plot.
points	A vector of integers. It should contain the ntID of the nucleotides to print in a different color.
colpoints	A string with the desired color if points is not NULL.
file	A string with the name of the output file. If NULL, the plot will be printed to screen.
width	The width of the plot (passed to the png() function).
height	The height of the plot (passed to the png() function).
bg	The background color of the plot (passed to the png() function).
units	The unit to measure height and width (passed to the png() function).
res	Resolution (passed to the png() function).
defaultview	A string to set different default options. Chose between '2Dupview', 'leftview' and 'rightview'.
thetaplot	Argument to be pased to persp3D() to set the view perspective.
phiplot	Argument to be pased to persp3D() to est the view perspective.
cleanerview	A logical to remove the lower density region to get a cleaner plot.

**Value**

A plot in screen, which can be directly saved to a ".png" file. \* plot2D A scatter plot. \* plot3Ddens A density map of the data in 3D.

**Author(s)**

Diego Gallego

**Examples**

```
ntinfo <- pipeNucData("1bau")
C3endo_ntID <- cleanByPucker(ntinfo, pucker="C3'endo")
plot2D(ntinfo=ntinfo, ntID=C3endo_ntID)
```

---

plotCategorical	<i>Barplot wrapper</i>
-----------------	------------------------

---

**Description**

Function to make more straightforward the process of plotting a barplot for categorical data.

**Usage**

```
plotCategorical(ntinfo, field, ntID = NULL, na.rm = FALSE,
  main = NULL, cex = 0.5, file = NULL, width = 15, height = 15,
  bg = "white", units = "cm", res = 200)
```

**Arguments**

ntinfo	A data.frame with the input data. It should contain the columns with the desired categorical data and a column labeled ntID.
field	The column name with the desired data.
ntID	A vector of integers with the desired nucleotides of analysis. If NULL all the nucleotides in the data.frame will be used.
na.rm	A logical to remove missing data.
main	A string with the title of the plot.
cex	To be passed to the par() function.
file	A string with the name of the output file. If NULL, the plot will be printed to screen.
width	The width of the plot (passed to the png() function).
height	The height of the plot (passed to the png() function).
bg	The background color of the plot (passed to the png() function).
units	The unit to measure height and width (passed to the png() function).
res	Resolution (passed to the png() function).

**Value**

A barplot with the categorical data of interest, which can be directly saved to a ".png" file.

**Author(s)**

Diego Gallego

**Examples**

```
## To see all the types of trinucleotides in the dataset:
ntinfo <- pipeNucData("1bau")
plotCategorical(ntinfo=ntinfo, field="localenv")
```

---

plotCircularDistribution

*Plot a scatter&frequency circular plot for angular data*

---

**Description**

For a vector of angular data (0 to 360), the function plots the distribution in a circular format.

**Usage**

```
plotCircularDistribution(data, clockwise = FALSE, start.degree = 0,
  main = NULL)
```

**Arguments**

data	A numeric vector with the data to plot.
clockwise	A logical indicating the sense in which the data should be displayed.
start.degree	An integer with the position in which the data starts being plotted.
main	A string with the title of the plot.

**Value**

A circular plot with the input data.

**Author(s)**

Diego Gallego

**Examples**

```
ntinfo <- pipeNucData("1bau")
C3endo_ntID <- cleanByPucker(ntinfo, pucker="C3'endo")
C3endo <- ntinfo[ntinfo$ntID %in% C3endo_ntID, ]
plotCircularDistribution(C3endo[, "delta"])
```

---

queryAPI

*Launch queries to the MMB or EBI APIs*


---

**Description**

Given a 4-character string (PDB ID) and the desired "info", it sends a query to the desired API and returns the output. This is an intermediate wrapper called by most of the queryFunctions (for documentation see ?queryFunctions).

**Usage**

```
queryAPI(ID, info = NULL, API = "default", string1 = NULL,
  string2 = NULL, reuse = TRUE, envir = parent.frame(n = 2),
  verbose = FALSE, noerrors = FALSE)
```

**Arguments**

ID	A 4 character string that matches a structure in the Protein Data Bank, or a 3 character string matching a compound.
info	A string with the desired query name.
API	A string that matches "ebi" or "mmb".
string1	A string to configure the query. See example below.
string2	A string to configure the query. See example below.



reuse	A logical. Set to TRUE if the same query is going to be send repeatedly, so that the result is saved in RAM (ir provides faster user access and avoids unnecessary work in the servers).
envir	Environment to save&retrieve the data if reuse is TRUE.
verbose	A logical. TRUE to print details of the process.
noerrors	A logical. TRUE to suppress errors.

**Value**

A vector or data.frame with the desired data.

**Author(s)**

Diego Gallego

**Examples**

```
## Imagine you want to programmatically access the EBI API contents
## through "http://www.ebi.ac.uk/pdbe/api/topology/entry/1s72/chain/H".
## 'queryAPI' understands it with four intructions:
## 'API="ebi"' stands for the root of the website name ("http.../api/").
## 'string1' is the string from the root to the pdb ID.
## 'ID' is just the PDB code.
## 'string2' is the string after the pdb ID.
## Thus, the call would be:
#data <- queryAPI(ID="1s72", API="ebi",
#                 string1="topology/entry/", string2="chain/H/")
```

---

queryEntryList

---

*Downloads the list of ID of ALL current PDB entries*


---

**Description**

Function to get the list of ALL PDB IDs in the Protein Data Bank at the moment.

**Usage**

```
queryEntryList(justIDs = TRUE)
```

**Arguments**

justIDs            A logical to return only pdbIDs without other information.

**Value**

A vector with all the PDB ID entries (updated weekly).

**Author(s)**

Diego Gallego

**Examples**

```
# pdblast <- queryEntryList()
```

---

queryFunctions

*General functions to query PDB (Protein Data Bank) data*

---

**Description**

Strightforward way to access structural data by making queries through the EBI or MMB mirrors of the PDB.

**Usage**

```
queryAuthors(pdbID, ...)  
queryChains(pdbID, chain = NULL, subset = NULL, ...)  
queryDescription(pdbID, ...)  
queryCompType(pdbID, ...)  
queryDepdate(pdbID, ...)  
queryEntities(pdbID, ...)  
queryFormats(pdbID, ...)  
queryHeader(pdbID, ...)  
queryHetAtms(pdbID, NAtNa = TRUE, ...)  
queryModres(pdbID, onlymodres = FALSE, ...)  
queryNDBId(pdbID, ...)  
queryLigands(pdbID, onlyligands = FALSE, NAtNa = TRUE, ...)  
queryOrgLigands(pdbID, ...)  
queryReldate(pdbID, ...)  
queryResol(pdbID, force = FALSE, ...)
```

```

queryRevdate(pdbID, ...)

queryStatus(pdbID, ...)

queryTechnique(pdbID, force = FALSE, ...)

```

### Arguments

pdbID	A 4-character string that matches a structure in the Protein Data Bank.
...	For advanced usage, arguments to be passed to subfunction <a href="#">queryAPI()</a> .
chain	A string with the chain identifier (in case you are only interested in a particular chain). If NULL, the info about all the chains is returned.
subset	Optional argument indicating "type", "length" or "description". If NULL, all the columns in the data.frame are returned.
NAtNa	A logical. If TRUE, sodium ion (NA) is modified as "Na".
onlymodres	A logical. If TRUE, only the modified residues are returned.
onlyligands	A logical. If TRUE, the function only returns the list of unique ligands.
force	A logical to force the query to the API (TRUE) or allow checking presaved data.

### Value

A character vector or data.frame with the desired information: \* queryAuthors List of authors. \* queryChains Data frame with list of chains and properties. \* queryDescription Author description of the entry. \* queryCompType Type of entry as defined in PDB (e.g. Prot-nuc). \* queryDepdate Deposition date. \* queryEntities Data frame with list of entities and properties. \* queryFormats Files available for the entry (e.g. to check if pdb format is available for the structure). \* queryHeader Classification of the structure as it appears in the header (PDB format) or in the "\_struct\_keywords.pdbx\_keywords" field (mmCIF format). \* queryHetAtms List of HETATM (modified residues and ligands). \* queryModres List of modified residues. \* queryNDBId NDB ID for Nucleic Acids. \* queryLigands Retrieves ligands. \* queryOrgLigands Retrieves just the organic ligands (not ions). \* queryReldate Release date. \* queryResol Resolution. \* queryRevdate Revision date. \* queryStatus PDB status. \* queryTechnique Experimental technique.

### Author(s)

Diego Gallego

### References

Official PDBe REST API site: <http://www.ebi.ac.uk/pdbe/pdbe-rest-api>  
 Official MMB API site: <http://mmb.irbbarcelona.org/api/>

### Examples

```

queryTechnique("4y4o")
queryAuthors("1s72")
queryNDBId("1bau")

```

---

queryObsoleteList	<i>Downloads the list of ID of ALL current PDB entries</i>
-------------------	--

---

**Description**

Function to get the list of Obsolete PDB IDs in the Protein Data Bank at the moment.

**Usage**

```
queryObsoleteList()
```

**Value**

A vector with all the PDB ID entries (updated weekly).

**Author(s)**

Diego Gallego

**Examples**

```
# obsolete <- queryObsoleteList()
```

---

represAsDataFrame	<i>Coerce Representative list to a data.frame</i>
-------------------	---

---

**Description**

Takes the output of `getRNAList` or `getAltRepres`, which represent molecules with the format "XXXXXIMIC+XXXXXIMIC" (XXXX: PDB ID; M: Model; C: Chain) and returns a `data.frame` with a more friendly structure:

- Col 1: Equivalence Class.
- Col 2: PDB ID.
- Col 3: Model.
- Col 4: Chain.

Columns 2 to 4 can be the direct input of `pipeNucData()`.

**Usage**

```
represAsDataFrame(nrlist)
```

**Arguments**

`nrlist`                      The output of `getRNAList()` or `getAltRepres()`.

**Value**

A data frame with the data of the representative structures.

**Author(s)**

Diego Gallego

**Examples**

```
data <- getRNAList(release=3.2, threshold="1.5A")
reps <- represAsDataFrame(nrlist=data)
```

---

RMSD

*Compute the RMSD between two structures*

---

**Description**

Compute the RMSD between two structures

**Usage**

```
RMSD(cif1, cif2, sel1 = NULL, sel2 = NULL, ...)

## S4 method for signature 'CIF,CIF'
RMSD(cif1 = NULL, cif2 = NULL, sel1 = NULL,
      sel2 = NULL, ...)

## S4 method for signature 'ANY,ANY'
RMSD(cif1 = NULL, cif2 = NULL, sel1 = NULL,
      sel2 = NULL, ...)
```

**Arguments**

cif1	A CIF or pdb object as obtained from <code>cifParser()</code> , <code>cifAsPDB()</code> or <code>read.cif/read.pdb</code> (from <code>bio3d</code> package).
cif2	A CIF or pdb object as obtained from <code>cifParser()</code> , <code>cifAsPDB()</code> or <code>read.cif/read.pdb</code> (from <code>bio3d</code> package).
sel1	A sel object as obtained from <code>atom.select</code> (from <code>bio3d</code> package).
sel2	A sel object as obtained from <code>atom.select</code> (from <code>bio3d</code> package).
...	Arguments to be passed to <code>atom.select</code> if desired (only used in case that <code>sel1</code> and <code>sel2</code> are <code>NULL</code> ).

**Value**

A numeric with the RMSD between the two structures.

**Author(s)**

Diego Gallego

**Examples**

```

cif <- cifParser("2d18")
model1 <- selectModel(cif=cif, model=1)
model3 <- selectModel(cif=cif, model=3)
RMSD(cif1=model1, cif2=model3)

```

rVector

*Compute the rVectors between the bases of a RNA structure***Description**

Given a RNA structure it computes the "r" vectors between all bases (Bottaro et al. 2014). This function is basic to compute the epsilon RMSD.

Furthermore, it also computes two more metrics:

1. The gamma angle, as obtained between the x axis of the coordinate system for all the bases projected on the reference base plane and the x axis of the latter. This is a metric of the relative rotation between bases along the orthogonal to the base plane axis (z).
2. The beta angle, as obtained between the base planes normal to account for the degree of coplanarity between bases.

**Usage**

```
rVector(cif, pdb, outformat = "rvector", simple_out = TRUE)
```

```
## S4 method for signature 'CIF'
```

```
rVector(cif, outformat = "rvector", simple_out = TRUE)
```

```
## S4 method for signature 'character'
```

```
rVector(cif, outformat = "rvector",
        simple_out = TRUE)
```

```
## S4 method for signature 'ANY'
```

```
rVector(pdb, outformat = "rvector", simple_out = TRUE)
```

**Arguments**

**cif** A CIF object as obtained from cifParser.

**pdb** A pdb object as obtained from cifAsPDB or read.cif/read.pdb (from bio3d package).

outformat      A string indicating the output format. This could be: "rvector", "vector\_coord" or "cylindrical\_coord".  
                  "rvector": (r(x)/a, r(y)/a, r(z)/b), being a=5 and b=3.  
                  "vector\_coord": (r(x), r(y), r(z)).  
                  "cylindrical\_coord": (rho, phy, z).  
                  See reference paper for more details. This does not apply to the gamma and beta angles.

simple\_out      A logical to simplify the output to a matrix.

### Value

A list of data.frames for the values of each base or a single matrix with all the data appended.

### Author(s)

Diego Gallego & Leonardo Darre

### References

Bottaro, S., F. Di Palma, and G Bussi. 2014. "The Role of Nucleobase Interactions in RNA Structure and Dynamics." *Nucleic Acids Research* 42 (21): 1330614

### Examples

```
cif <- cifParser("2d18")
model1 <- selectModel(cif=cif, model=1)
vectors <- rVector(cif=model1, simple_out=FALSE)
```

---

selectModel

*Selects a desired model from a CIF/pdb structure*

---

### Description

Given a object obtained from cifParser/cifAsPDB or read.cif/read.pdb functions (from bio3d specifying "multi = TRUE"), the function returns an object of the same type (CIF or pdb) with the desired model. Since some structures deposited in the PDB contain models with different number of atoms, the pdb objects in R require a special treatment. The cifAsPDB and selectModel functions can cope with these structures (e.g. 1JTW).

### Usage

```
selectModel(cif, pdb, model, verbose = FALSE)
```

```
## S4 method for signature 'CIF'
selectModel(cif, model, verbose = FALSE)
```

```
## S4 method for signature 'ANY'
selectModel(pdb, model, verbose = FALSE)
```

**Arguments**

cif	A CIF object as obtained from cifParser.
pdb	A pdb object with multiple models (obtained from cifAsPDB or read.cif/read.pdb from bio3d package).
model	A string with the desired model number.
verbose	A logical to print messages on screen.

**Value**

A CIF/pdb object with the desired model coordinates.

**Author(s)**

Diego Gallego

**Examples**

```
cif <- cifParser("1qfq")
model3 <- selectModel(cif=cif, model=3)
```

---

trimByID

---

*Calls trimSphere to generate smaller pdb files*


---

**Description**

Given a data frame with nucleotide info (as obtained from pipeNucData) and the desired nucleotide index (ntID), the function returns a pdb object or file allowing the user to select a number of 5' and 3' neighbors in sequence and non-connected residues in a cutoff radius.

**Usage**

```
trimByID(cif = NULL, ntID, ntinfo, prev = 2, post = 2,
         verbose = TRUE, file = NULL, justbackbone = FALSE, ...)
```

**Arguments**

cif	A cif/pdb object obtained from cifParser/read.pdb respectively or a pdb ID so that the function can download the data. If NULL, the function will extract the pdb ID from the ntinfo data frame (pdbID col).
ntID	An integer/string with the desired nucleotide ID for analysis.
ntinfo	a data.frame with the data. It should contain at least the columns "pdbID", "chain", "model", "resno", "insert" and "ntID" (as the output of pipeNucData function).
prev	Number of desired 5' neighbours to be returned.



post	Number of desired 3' neighbours to be returned.
verbose	A logical to print details of the process.
file	A string to save the output in a pdb formatted file. If NULL the functions just returns the pdb object.
justbackbone	A logical to keep only the backbone of the output pdb.
...	Arguments to be passed to trimSphere (type ?trimSphere for details).

**Value**

A smaller pdb object or a pdb file.

**Author(s)**

Diego Gallego

**Examples**

```
cif <- cifParser("1bau")
ntinfo <- pipeNucData(cif, torsionals=NULL, distances=NULL, angles=NULL)

## Obtain a smaller pdb of the 4th nucleotide +-2 neighbours and a
## surrounding sphere of 5 Angstroms
pdb <- trimByID(cif=cif, ntinfo=ntinfo, ntID=4, prev=2, post=2,
               cutoff=5)

## Same process saving the output in a file:
trimByID(cif=cif, ntinfo=ntinfo, ntID=4, prev=2, post=2,
        cutoff=5, file="output.pdb")
```

---

trimSphere	<i>Trim a pdb/cif object to obtain a nucleotide/s of interest and the surrounding area.</i>
------------	---

---

**Description**

From a pdb/CIF object, the nucleotide of interest and a radius, the function finds all the atoms in the given area and returns a pdb object that only includes the nearest atoms.

**Usage**

```
trimSphere(cif, model = NULL, ntindex = NULL, chain = NULL,
           sel = NULL, cutoff = 8, cutres = FALSE, file = NULL,
           verbose = TRUE, alt = "uniq", rename = TRUE, ...)
```

**Arguments**

cif	A cif/pdb object obtained from cifParser/read.pdb respectively or a pdb ID so that the function can download the data.
model	The model of interest to use in the calculations. The first model is always the default.
ntindex	A numeric index/indices for the position of the desired nucleotides in the given chain. Not necessary if you provide sel (see below).
chain	A string with the chain of interest. Not necessary if you provide sel (see below).
sel	A "select" object as obtained from atom.select (bio3d). Note that if you are using this option, cif must be the same input object you used for the atom.select function.
cutoff	A numeric indicating the radius in angstroms to select around the desired nucleotides. If 0 only the nucleotides are returned.
cutres	A logical. TRUE to return only what it is found in the cutoff (residues in the boundaries of the cutoff are usually truncated) or FALSE to return whole residues even if further than the cutoff.
file	A string to save the output in a pdb formatted file. If NULL the functions just returns the pdb object.
verbose	A logical to print details of the process.
alt	Value of alternative records to keep in output object.
rename	A logical to rename the chains of the output pdb.
...	Arguments to be passed to internal functions.

**Value**

A smaller pdb object or a pdb file.

**Author(s)**

Diego Gallego

**Examples**

```
## Toy example:
cif <- cifParser("1s72")

## Generate a smaller pdb with the residues 55 to 58 of the RNA chain
## "9" with a surrounding sphere of 5 Angstroms:
smallerpdb <- trimSphere(cif, ntindex=seq(55, 58, 1), chain="9",
                        cutoff=5, verbose=FALSE)

## Same process saving the output in a file:
smallerpdb <- trimSphere(cif, ntindex=seq(55, 58, 1), chain="9",
                        cutoff=5, verbose=FALSE, file="output.pdb")

## Second example:
```

```
## Obtain a PDB with just the interacting region between RNA and prot
#pdb <- cifAsPDB("1nyb")
#data <- findBindingSite(pdb, select="RNA", byres=TRUE)
#sel <- bio3d::atom.select(pdb,
#                           eleno=append(data$eleno_A, data$eleno_B))
#trimSphere(pdb, sel=sel, file="interacting_site.pdb", verbose=FALSE)
```

---

veriNA3d*veriNA3d: Structural bioinformatics package for data mining of the PDB.*

---

### Description

VeriNA3d is an R package for the analysis of Nucleic Acid structural data. The software was developed on top of bio3d (Grant et al, 2006) with a higher level of abstraction. In addition of single-structure analyses, veriNA3d also implements pipelines to handle whole datasets of mmCIF/PDB structures. As far as we know, no similar software has been previously distributed, thus it aims to fill a gap in the data mining pipelines of PDB structural data analyses.

### Author(s)

Diego Gallego [diego.gallego@irbbarcelona.org](mailto:diego.gallego@irbbarcelona.org)

# Index

## \*Topic **datasets**

- dataset\_wadley2007, [13](#)
- entities, [15](#)
- fastquery, [17](#)
- applyToPDB, [2](#)
- bio3d::read.pdb(), [14](#)
- checkNuc, [3](#)
- CIF (CIF-class), [4](#)
- CIF-class, [4](#)
- cif\_accessors, [8](#)
- cifAsPDB, [5](#)
- cifAsPDB(), [14](#), [23](#), [37](#)
- cifAsPDB, character-method (cifAsPDB), [5](#)
- cifAsPDB, CIF-method (cifAsPDB), [5](#)
- cifAtom\_site (cif\_accessors), [8](#)
- cifAtom\_site, CIF-method (cif\_accessors), [8](#)
- cifAtom\_sites (cif\_accessors), [8](#)
- cifAtom\_sites, CIF-method (cif\_accessors), [8](#)
- cifAtom\_type (cif\_accessors), [8](#)
- cifAtom\_type, CIF-method (cif\_accessors), [8](#)
- cifAudit\_author (cif\_accessors), [8](#)
- cifAudit\_author, CIF-method (cif\_accessors), [8](#)
- cifAudit\_conform (cif\_accessors), [8](#)
- cifAudit\_conform, CIF-method (cif\_accessors), [8](#)
- cifChem\_comp (cif\_accessors), [8](#)
- cifChem\_comp, CIF-method (cif\_accessors), [8](#)
- cifDatabase\_2 (cif\_accessors), [8](#)
- cifDatabase\_2, CIF-method (cif\_accessors), [8](#)
- cifDownload, [6](#)
- cifEntity (cif\_accessors), [8](#)
- cifEntity, CIF-method (cif\_accessors), [8](#)
- cifEntry (cif\_accessors), [8](#)
- cifEntry, CIF-method (cif\_accessors), [8](#)
- cifExptl (cif\_accessors), [8](#)
- cifExptl, CIF-method (cif\_accessors), [8](#)
- cifParser, [7](#)
- cifParser(), [5](#), [16](#), [37](#)
- cifParser, ANY-method (cifParser), [7](#)
- cifPdbx\_database\_status (cif\_accessors), [8](#)
- cifPdbx\_database\_status, CIF-method (cif\_accessors), [8](#)
- cifStruct (cif\_accessors), [8](#)
- cifStruct, CIF-method (cif\_accessors), [8](#)
- cifStruct\_asym (cif\_accessors), [8](#)
- cifStruct\_asym, CIF-method (cif\_accessors), [8](#)
- cifStruct\_keywords (cif\_accessors), [8](#)
- cifStruct\_keywords, CIF-method (cif\_accessors), [8](#)
- classifyDNA (classifyNA), [10](#)
- classifyNA, [10](#)
- classifyRNA (classifyNA), [10](#)
- cleanByPucker, [12](#)
- countEntities, [13](#)
- dataset\_wadley2007, [13](#)
- dssr, [14](#)
- entities, [15](#)
- eRMSD, [16](#)
- eRMSD, ANY, ANY-method (eRMSD), [16](#)
- eRMSD, CIF, CIF-method (eRMSD), [16](#)
- fastquery, [17](#)
- findBindingSite, [18](#)
- findBindingSite(), [28](#)
- findHDR, [19](#)
- getAltRepres, [20](#)

getAltRepres(), 36  
getRNAList, 21  
getRNAList(), 36  
  
hasHetAtm, 22  
  
measureElenoDist, 22  
measureElenoDist(), 24  
measureEntityDist, 24  
measureNuc, 25  
measureNuc(), 27  
  
pipeNucData, 26  
pipeNucData(), 12, 28, 36  
pipeProtNucData, 28  
plot2D, 29  
plot3Ddens (plot2D), 29  
plotCategorical, 30  
plotCircularDistribution, 31  
  
queryAPI, 32  
queryAPI(), 35  
queryAuthors (queryFunctions), 34  
queryChains (queryFunctions), 34  
queryCompType (queryFunctions), 34  
queryDepdate (queryFunctions), 34  
queryDescription (queryFunctions), 34  
queryEntities (queryFunctions), 34  
queryEntryList, 33  
queryFormats (queryFunctions), 34  
queryFunctions, 34  
queryHeader (queryFunctions), 34  
queryHetAtms (queryFunctions), 34  
queryHetAtms(), 22  
queryLigands (queryFunctions), 34  
queryModres (queryFunctions), 34  
queryNDBId (queryFunctions), 34  
queryObsoleteList, 36  
queryOrgLigands (queryFunctions), 34  
queryReldate (queryFunctions), 34  
queryResol (queryFunctions), 34  
queryRevdate (queryFunctions), 34  
queryStatus (queryFunctions), 34  
queryTechnique (queryFunctions), 34  
  
represAsDataFrame, 36  
RMSD, 37  
RMSD, ANY, ANY-method (RMSD), 37  
RMSD, CIF, CIF-method (RMSD), 37  
  
rVector, 38  
rVector(), 16  
rVector, ANY-method (rVector), 38  
rVector, character-method (rVector), 38  
rVector, CIF-method (rVector), 38  
  
selectModel, 39  
selectModel, ANY-method (selectModel), 39  
selectModel, CIF-method (selectModel), 39  
  
trimByID, 40  
trimSphere, 41  
  
veriNA3d, 43  
veriNA3d-package (veriNA3d), 43