Package 'vanddraabe'

October 12, 2022

Type Package

Title Identification and Statistical Analysis of Conserved Waters Near Proteins

Description Identify and analyze conserved waters within crystallographic protein structures and molecular dynamics simulation trajectories. Statistical parameters for each water cluster, informative graphs, and a PyMOL session file to visually explore the conserved waters and protein are returned. Hydrophilicity is the propensity of waters to congregate near specific protein atoms and is related to conserved waters. An informatics derived set of hydrophilicity values are provided based on a large, high-quality X-ray protein structure dataset.

Version 1.1.1

Date 2019-06-10

Depends R (>= 3.6.0)

Imports bio3d (>= 2.3-4), cowplot (>= 0.9.4), fastcluster (>= 1.1.25), ggplot2 (>= 3.1.1), openxlsx (>= 4.1.0), reshape2 (>= 1.4.3), scales (>= 1.0.0)

Suggests knitr, rmarkdown, testthat

URL http://vanddraabe.com, https://github.com/exeResearch/vanddraabe/

BugReports https://github.com/exeResearch/vanddraabe/issues

License MIT + file LICENSE

LazyData TRUE

RoxygenNote 6.1.1

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Emilio Xavier Esposito [aut, cre]

Maintainer Emilio Xavier Esposito <emilio@exeResearch.com>

Repository CRAN

Date/Publication 2019-06-07 22:00:03 UTC

R topics documented:

`aaStandardizeNames` *Standardize Amino Acid Names*

Description

Standardize the various three-letter amino acid residue names.

Usage

```
aaStandardizeNames(residue.names)
```

Arguments

`residue.names`	A vector of strings containing the three-letter residue names (strings)

Details

The various three-letter amino acid residue names used to indicate protonation state or uncommon sidechain bonding (ligatation) are converted to the standard amino acid residue name.

Value

vector of *standardized* amino acid residue names

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: `ConservationSet`, `DetermineChainsOfInterest`, `ExtractFileTimeStamp`, `ExtractPDBids`, `FileTimeStamp`, `HasXWaters`, `Nearby`, `ProtHetWatIndices`, `RescaleValues`, `RetainChainsOfInterest`, `ReturnPDBfullPath`, `StandardizeAsparticAcidNames`, `StandardizeCysteineNames`, `StandardizeGlutamicAcidNames`, `StandardizeHistidineNames`, `StandardizeLysineNames`, `TimeSpan`, `UniqueAtomHashes`, `getAtomTypeCounts`, `getResTypeCounts`, `res2xyz`, `resAtomType2AtomClass`, `write.basic.pdb`, `write.conservedWaters.pdb`

Examples

```
residue.names <- c("HIS", "HID", "HIE", "HIP", "HSD", "HSE", "HSP",
                   "CYS", "CYM", "CYX", "ASP", "ASH", "GLU", "GLH",
                   "LYS", "LYN")
aaStandardizeNames(residue.names)
# [1] "HIS" "HIS" "HIS" "HIS" "HIS" "HIS" "HIS" "CYS" "CYS" "CYS"
#     "ASP" "ASP" "GLU" "GLU" "LYS" "LYS"
```

AlignOverlap *Alignment Overlap Check*

Description

Determine if two protein structures are aligned using C-alpha atoms.

Usage

```
AlignOverlap(aligned.dir = "blast_fitlsq_1.0ang", out.dir = "blast",
  ref.PDBid = "1hai", overlap = 0.7, removal = 0.1, CA.dist = 1.25,
  filename = "ProteinSystem")
```

Arguments

aligned.dir	Directory with aligned structures
out.dir	Directory prefix for the correctly and incorrectly aligned structures. The out.dir variable is also used to construct the Excel workbook with the summary of the alignment evaluation
ref.PDBid	Reference structure PDB ID, four character ID, used to compare all other aligned structures
overlap	The ratio of overlapping C-alpha atoms; default 0.70
removal	The ratio of overlapping C-alpha atoms to remove a chain from a collection of chains passing the overlap requirement; default: 0.10
CA.dist	The minimum distance between C-alpha atoms for the two C-alpha atoms to be considered aligned; default: 1.25
filename	The filename of the Excel workbook containing all the results from the analysis.

Details

Using the C-alpha atoms of two aligned proteins, the amount of atomic overlap is determined and the overlapped chains are written to individual PDB files in the NAME_alignedGood directory. The PDB files have the PDBID_aligned_pruned.pdb naming convention where the PDBID is the RCSB four-character identification code. Structures not meeting the user defined overlap ratio are written to the NAME_alignedPoor directory. The structures are written using the bio3d::write.pdb() function of the bio3d package.

Value

This function returns:

- **Overlapping sturctures**: PDB structures satisfying the overlap requirements are written to the out.dir_alignedGood directory
- **Non-Overlapping sturctures**: PDB structures *not* satisfying the overlap requirement are written to out.dir_alignedPoor
- **AlignOverlap.summary**: data.frame of the information written to the Excel workbook
- **call**: The user provided parameters for the function

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Alignment Overlap": CalcAlignOverlap

Examples

```
 ## Not run:
  ## example from the Thrombin vignette
  AlignOverlap(aligned.dir = "",
               out.dir = "OVERLAP",
               ref.PDBid = "1hai",
               overlap = 0.70, removal = 0.10,
               CA.dist = 1.25,
               filename = "Thrombin")

## End(Not run)
```

BoundWaterEnvironment *Bound Water Environment*

Description

Various enviroment counts for bound waters.

Usage

```
BoundWaterEnvironment(distances, set.oi.idc, names.atoms, names.res.atoms,
  structure, radius = 3.6)
```

Arguments

distances	Matrix of atomic pairwise distances
set.oi.idc	Indices of atoms of interest; can be protein, water, or HETATMs if those are of interest
names.atoms	Atom names for the atoms of interest. Valid atom names are provided in the names.backbone.atoms() and names.sidechain.atoms() functions; *e.g.*; "C", "O", "CB", "OG1", "CG2", "N"
names.res.atoms	Residue and atom names of interest. Valid residue-atom names are provided in the names.res.AtomTypes() function; *e.g.*; "THR C", "THR O", "THR CB", "THR OG1"
structure	The protein structure of interest with its residue and atom names; X, Y, and Z coordinates; residue and atom numbers; and B-value, Normalized B-value, Occupancy, and Mobility values.
radius	Distance in Angstroms between the atoms of interest; default: 3.6 Angstroms

Details

For the heavy atoms near each water molecule (oxygen atom) the bound water environment is calculated. These values are defined in the **Return** section. The default radius distance is 3.6 Angstroms. While it is possible to define the radius to a value other than 3.6 this value is hardcoded into the `ConservedWaters()` function. This might change in future versions.

NOTE: This function is designed to work with `ConservedWaters()` via the `base::apply()` function processing rows (the `MARGIN = 1` option). For this reason it is **NOT** a public function. The `Nearby()` is specifically designed to work with this function.

Value

A list of the bound water environment values for nearby heavy atoms.

- **adn**: num of nearby heavy atoms
- **ahp.sum**: sum of hydrodrophilicy values
- **ahp.mu**: mean of hydrodrophilicy values
- **ahp.sd**: standard deviation of hydrodrophilicy values
- **hbonds**: number of possible hydrogen bonds
- **o.sum**: sum of occupancy values
- **o.mu**: mean of occupancy values
- **o.sd**: standard deviation of occupancy values
- **b.exp.sum**: sum of experimental B-values
- **b.exp.mu**: mean of experimental B-values
- **b.exp.sd**: standard deviation of experimental B-values
- **mobility.sum**: sum of mobility values
- **mobility.mu**: mean of mobility values
- **mobility.sd**: standard deviation of mobility values
- **nBvalue.sum**: sum of normalized Bvalues
- **nBvalue.mu**: mean of normalized Bvalues
- **nBvalue.sd**: standard deviation of normalized Bvalues

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

Leslie A Kuhn, Craig A Swanson, Michael E Pique, John A Tainer, and Elizabeth D Getzof. Atomic and Residue Hydrophilicity in the Context of Folded Protein Structures. *PROTEINS: Structure, Function, and Genetics*, 1995, **2** (*4*), pp 536-547. DOI: 10.1002/prot.340230408 PMID: 8749849

See Also

Other "Bound Water Environment": BoundWaterEnvironment.interact, BoundWaterEnvironment.quality, Mobility, NormalizedBvalue, calcBvalue, calcNearbyHydrationFraction, calcNumHydrogenBonds

Examples

```
## Not run:
distances <- PDB.1hai.h2o.prot.dists[3, ]
set.oi.idc <- prot.idc
names.atoms <- PDB.1hai.aoi.clean$elety[prot.idc]
names.res.atoms <- paste(PDB.1hai.aoi.clean$resid[prot.idc], names.atoms, sep =" ")
structure <- PDB.1hai.aoi.clean
BoundWaterEnvironment(distances,
                      set.oi.idc,
                      names.atoms,
                      names.res.atoms,
                      structure,
                      radius = 3.6)

## End(Not run)
```

BoundWaterEnvironment.interact

Bound Water Environment (interactions)

Description

Various enviroment counts for bound waters.

Usage

```
BoundWaterEnvironment.interact(distances, set.oi.idc, names.atoms,
  names.res.atoms, radius = 3.6)
```

Arguments

distances	Matrix of atomic pairwise distances
set.oi.idc	Indices of protein atoms; can also HETATMs if those are of interest
names.atoms	Atom names from the PDB file in the PDB atomic naming convention.
names.res.atoms	Atom names of the form "RES AT"; created by combining the residue and atom name while separating the two by a space. These do not need to be unique because these names will be used to lookup hydrophilicity values.
radius	Distance in Angstroms between the atoms of interest; default: 3.6 Angstroms

Details

For the heavy atoms near each water molecule (oxygen atom) the bound water environment is calculated. These values are defined in the **Return** section. The default radius distance is 3.6 Angstroms. While it is possible to define the radius to a value other than 3.6 this value is hardcoded into the `ConservedWaters()` function. This might change in future versions.

NOTE: This function is designed to work with `ConservedWaters()` via the `base::apply()` function processing rows (the `MARGIN = 1` option). For this reason it is **NOT** a public function. The `Nearby()` is specifically designed to work with this function.

Value

A list of the bound water environment values for nearby heavy atoms.

- **adn**: num of nearby heavy atoms
- **ahp.sum**: sum of hydrodrophilicy values
- **ahp.mu**: mean of hydrodrophilicy values
- **ahp.sd**: standard deviation of hydrodrophilicy values
- **hbonds**: number of possible hydrogen bonds

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

Leslie A Kuhn, Craig A Swanson, Michael E Pique, John A Tainer, and Elizabeth D Getzof. Atomic and Residue Hydrophilicity in the Context of Folded Protein Structures. *PROTEINS: Structure, Function, and Genetics*, 1995, **2** (*4*), pp 536-547. DOI: 10.1002/prot.340230408 PMID: 8749849

See Also

Other "Bound Water Environment": `BoundWaterEnvironment.quality`, `BoundWaterEnvironment`, `Mobility`, `NormalizedBvalue`, `calcBvalue`, `calcNearbyHydrationFraction`, `calcNumHydrogenBonds`

Examples

```
## Not run:
distances <- PDB.1hai.h2o.prot.dists[3, ]
set.oi.idc <- prot.idc
names.atoms <- PDB.1hai.aoi.clean$elety[prot.idc]
names.res.atoms <- paste(PDB.1hai.aoi.clean$resid[prot.idc], names.atoms, sep =" ")
BoundWaterEnvironment.interact(distances,
                               set.oi.idc,
                               names.atoms,
```

```
                                  names.res.atoms,
                                  radius = 3.6)
  # $adn
  # [1] 9
  #
  # $ahp.sum
  # [1] 2.001
  #
  # $ahp.mu
  # [1] 0.2223
  #
  # $ahp.sd
  # [1] 0.2229
  #
  # $hbonds
  # [1] 4

## End(Not run)
```

`BoundWaterEnvironment.quality` *Bound Water Environment (atomic quality)*

Description

Various enviroment counts for bound waters.

Usage

```
BoundWaterEnvironment.quality(distances, set.oi.idc, structure,
  radius = 3.6)
```

Arguments

`distances`	Matrix of atomic pairwise distances
`set.oi.idc`	Indices of atoms of interest; can be protein, water, or HETATMs if those are of interest
`structure`	The protein structure of interest with its residue and atom names; X, Y, and Z coordinates; residue and atom numbers; and B-value, Normalized B-value, Occupancy, and Mobility values.
`radius`	Distance in Angstroms between the atoms of interest; default: 3.6 Angstroms

Details

For the heavy atoms near each water molecule (oxygen atom) the bound water environment is calculated. These values are defined in the **Return** section. The default radius distance is 3.6 Angstroms. While it is possible to define the radius to a value other than 3.6 this value is hardcoded into the `ConservedWaters()` function. This might change in future versions.

NOTE: This function is designed to work with `ConservedWaters()` via the `base::apply()` function processing rows (the `MARGIN = 1` option). For this reason it is **NOT** a public function. The `Nearby()` is specifically designed to work with this function.

Value

A list of the bound water environment values for nearby heavy atoms.

- **o.sum**: sum of occupancy values
- **o.mu**: mean of occupancy values
- **o.sd**: standard deviation of occupancy values
- **b.exp.sum**: sum of experimental B-values
- **b.exp.mu**: mean of experimental B-values
- **b.exp.sd**: standard deviation of experimental B-values
- **mobility.sum**: sum of mobility values
- **mobility.mu**: mean of mobility values
- **mobility.sd**: standard deviation of mobility values
- **nBvalue.sum**: sum of normalized B-values
- **nBvalue.mu**: mean of normalized B-values
- **nBvalue.sd**: standard deviation of normalized B-values

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

Leslie A Kuhn, Craig A Swanson, Michael E Pique, John A Tainer, and Elizabeth D Getzof. Atomic and Residue Hydrophilicity in the Context of Folded Protein Structures. *PROTEINS: Structure, Function, and Genetics*, 1995, **2** (*4*), pp 536-547. DOI: 10.1002/prot.340230408 PMID: 8749849

See Also

Other "Bound Water Environment": `BoundWaterEnvironment.interact`, `BoundWaterEnvironment`, `Mobility`, `NormalizedBvalue`, `calcBvalue`, `calcNearbyHydrationFraction`, `calcNumHydrogenBonds`

Examples

```
## Not run:
distances <- PDB.1hai.h2o.prot.dists[3, ]
set.oi.idc <- prot.idc
structure <- PDB.1hai.aoi.clean
BoundWaterEnvironment.quality(distances,
                              set.oi.idc,
                              structure,
                              radius = 3.6)

## End(Not run)
```

`BoundWaterEnvPlots` *Bound Water Environment Barplots*

Description

Normalized B-value Barplots for Cluster with at least 50% Conservation

Usage

```
BoundWaterEnvPlots(data, passed.waters = TRUE, pct.conserved.gte = 50,
  num.clusters = 50)
```

Arguments

`data`	The `h2o.clusters.summary` data.frame from the `ClusterWaters` function containing the `nBvalue.mu` information.
`passed.waters`	Logical indicator to plot results for waters **passing** `Mobility()` and `NormalizedBvalue()` **OR** using **all** waters within the `PDB` files.
`pct.conserved.gte`	minimum percent conservation within a water cluster; default: `50.0`If the number of clusters is less than the number of clusters defined by `num.clusters`, then the number of clusters defined by `pct.conserved.gte` is displayed.
`num.clusters`	number (integer) of clusters to display. If the number of clusters is less than the number of clusters defined by `pct.conserved.gte`, then the number of clusters defined by `num.clusters` is displayed. A value of `NULL` results in the provided value for `pct.conserved.gte` being used.

Details

Constructs a barplot with corresponding density plot for the mean normalized B-value value for all water within each cluster with at least 50% water conservation. Clusters with 50 to 69% water conseration are dark red, clusters with 70 to 79% conservation are red, 80 to 89% conservation are light blue, 90 to 99% conservation are blue, and 100% conservation (waters from all structures) are dark blue.

The normalized B-value values are calculated by the NormalizedBvalue function.

This plot was inspired by Figure 1 of Sanschagrin and Kuhn (1998).

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

See Also

Other plots: BoundWaterEnvSummaryPlot, BvalueBarplot.summ, BvalueBarplot, ClusterSummaryPlots, MobNormBvalEvalPlots, MobilityBarplot.summ, MobilityBarplot, OccupancyBarplot.summ, OccupancyBarplot, nBvalueBarplot, normBvalueBarplot.summ

Examples

```
## Not run:
bwe.plots <- BoundWaterEnvPlots(data=thrombin10.conservedWaters,
                                passed.waters=TRUE,
                                pct.conserved.gte = 50.0,
                                num.clusters = 50)

## End(Not run)
```

BoundWaterEnvSummaryPlot

Bound Water Environment Summary Plot

Description

Mean bound water environment summary per percent conservation

Usage

```
BoundWaterEnvSummaryPlot(data, passed.waters = TRUE,
  title = "Bound Water Environment per Conservation")
```

Arguments

`data`	The `h2o.clusters.summary` data.frame from the `ClusterWaters` function containing the `nBvalue.mu` information. This data.frame is found within the `h2o.cluster.passed` and `h2o.cluster.all`
`passed.waters`	Logical indicator to plot results for waters **passing** `Mobility()` and `NormalizedBvalue()` **OR** using **all** waters within the `PDB` files.
`title`	The title for the plot

Details

Constructs a line plot with the bound water environment measures for the nearby protein and water atoms. The protein atomic density (ADN), hydrophilicity, mobility, normalized B-values, and potential hydrogen bonds are summarized for protein heavy atoms with 3.6 Angstroms along with the mobility, normalized B-values, and hydrogen bonds are summarized for the waters within 3.6 Angstroms of the protein and water atoms of interest, respectively. The raw values are scaled to values between 0 and 1 and plotted for each of the percent conservation available. Thus if there are ten structures being analyzed the percent conservation can range from 10 to 100% in 10% increments. The protein related values are shown as solid lines and the water related values are shown as dotted lines.

Interpreting the plot

- **dark green**: protein atom density
- **medium green**: protein atom hydrophilicity
- **green**: protein mobility
- **pale green**: protein nBvalue
- **light green**: protein hydrogen bonds
- **dark blue**: water mobility
- **medium blue**: water nBvalue
- **blue**: water hydrogen bonds

This plot is based on Figure 3 of Sanschagrin and Kuhn (1998). Please note the B-value have been replaced with normalized B-values and hydrophilicity has been removed. Hydrophilicity was removed because the range between average hydrophilicity values for the percent conservations would likely be narrow. Due to the way scaling works, the lowest value is scaled to zero and the greatest value is scaled to one. Scaling the mean hydrophilicity values works against our goal of showing an overall tread and instead creates confusion about the values.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

See Also

Other plots: BoundWaterEnvPlots, BvalueBarplot.summ, BvalueBarplot, ClusterSummaryPlots, MobNormBvalEvalPlots, MobilityBarplot.summ, MobilityBarplot, OccupancyBarplot.summ, OccupancyBarplot, nBvalueBarplot, normBvalueBarplot.summ

Examples

```
## Not run:
bwe.summary.plot <- BoundWaterEnvSummaryPlot(data=thrombin10.conservedWaters,
                                             passed.waters=TRUE,
                                    title="Bound Water Environment per Conservation")

## End(Not run)
```

BvalueBarplot *B-value Barplots*

Description

B-value Barplots for Cluster with at least 50% Conservation

Usage

```
BvalueBarplot(data, passed.waters = TRUE, calc.values = TRUE)
```

Arguments

data	The h2o.clusters.summary data.frame from the ClusterWaters() function containing the b.exp.mu information.
passed.waters	Logical indicator to plot results for waters **passing** Mobility() and NormalizedBvalue() **OR** using **all** waters within the PDB files.
calc.values	Plot the calculated B-values the mean experimental B-values; default: TRUE

Details

Constructs a barplot with corresponding density plot for the mean B-value value for all water within each cluster with at least 50% water conservation. Clusters with 50 to 69% water conseration are dark red, clusters with 70 to 79% conservation are red, 80 to 89% conservation are light blue, 90 to 99% conservation are blue, and 100% conservation (waters from all structures) are dark blue.

This plot was inspired by Figure 1 of Sanschagrin and Kuhn (1998).

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

See Also

Other plots: BoundWaterEnvPlots, BoundWaterEnvSummaryPlot, BvalueBarplot.summ, ClusterSummaryPlots, MobNormBvalEvalPlots, MobilityBarplot.summ, MobilityBarplot, OccupancyBarplot.summ, OccupancyBarplot, nBvalueBarplot, normBvalueBarplot.summ

Examples

```
## Not run:
Bvalue.plot <- BvalueBarplot(data=thrombin10.conservedWaters,
                             passed.waters=TRUE)

## End(Not run)
```

BvalueBarplot.summ *B-value Summary Barplots*

Description

B-value summary barplots for the PDB structures. The plots are faceted and displays the binned B-value values for all the structures. The counts are presented on a log10 scale.

Usage

```
BvalueBarplot.summ(data)
```

Arguments

data	The results from the CleanProteinStructures() function. Will use the binned B-value data.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other plots: BoundWaterEnvPlots, BoundWaterEnvSummaryPlot, BvalueBarplot, ClusterSummaryPlots, MobNormBvalEvalPlots, MobilityBarplot.summ, MobilityBarplot, OccupancyBarplot.summ, OccupancyBarplot, nBvalueBarplot, normBvalueBarplot.summ

Examples

```
## Not run:
BvalueBarplot.summ(data)

##----- multiple pages
library(ggforce)
Bvalue.barplots.summary <- BvalueBarplot.summ(data)
num.pages <- ceiling(nrow(data$Bvalue.counts) / 10)

pdf(file="multiple_pages.pdf", height=11, width=8.5)
for (page in seq_len(num.pages)) {
  print(Bvalue.barplots.summary +
        ggforce::facet_wrap_paginate(~PDBid,
                                     ncol = 2, nrow = 5, page = page) )
}
dev.off()

## End(Not run)
```

`CalcAlignOverlap` *Calculate Alignment Overlap*

Description

Calculate the amount of alignment overlap between two protein structures using C-alpha atoms.

Usage

```
CalcAlignOverlap(ref.num.atoms, ref.ca, ref.idc, soi.PDB, CA.dist)
```

Arguments

`ref.num.atoms`	Number of atoms in the reference structure
`ref.ca`	PDB formatted `data.frame` containing only C-alpha atoms
`ref.idc`	The indices of the reference structure atoms; from 1 to the number of atoms in the reference structure
`soi.PDB`	The structure of interest (SoI) being compared to the reference structure. This is the full PDB structure read into R using the `bio3d::read.pdb2()` function
`CA.dist`	The minimum distance between C-alpha atoms for the two C-alpha atoms to be considered aligned; default: 1.25

Details

Using the C-alpha atoms of two aligned proteins, the amount of atomic overlap is determined. This function is within the AlignOverlap function.

This is a **non-public** function and is **NOT** available for general use. Please contact the author if you believe this function should be available for general use.

Value

This function returns:

- **ratio.intersection**: fraction of SOI overlapping with the reference structure
- **soi.chain**: Chain letter designations for the aligned SOI
- **soi.chain.overlap**: Unique chain letter designations for the aligned SOI These values are then used within the `AlignOverlap()` function to determine if the structures are adequately aligned.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Alignment Overlap": `AlignOverlap`

---

calcAtomClassHydrophilicity

Atom Class Hydration Fraction

Description

Calculates the mean hydration value for atoms within a class.

Usage

```
calcAtomClassHydrophilicity(df.AtomHydroTEMP)
```

Arguments

`df.AtomHydroTEMP`
: The newly calculated (determined) atomic hydrophilicity values

Details

This function is called within `HydrophilicityEvaluation()` to calculate the hydration fraction for the five atom classes listed in the *Value* section.

$$(numsurfaceexposures)/(numatomoccurrences)$$

NOTE: This is a non-public function.

Value

This function returns:

- **hydratFraction.oxy.neut**: Neutral oxygen atoms; enter `names.resATs.oxy.neut` to see list of residue-atomtypes
- **hydratFraction.oxy.neg**: Negative oxygen atoms; enter `names.resATs.oxy.neg` to see list of residue-atomtypes
- **hydratFraction.nitro.neut**: Neutral nitrogen atoms; enter `names.resATs.nitro.neut` to see list of residue-atomtypes
- **hydratFraction.nitro.pos**: Positive nitrogen atoms; enter `names.resATs.nitro.pos` to see list of residue-atomtypes
- **hydratFraction.carb.sulf**: Carbon and sulfur atoms; enter `names.resATs.carb.sulf` to see list of residue-atomtypes

These values are returned in `HydrophilicityValues.AtomTypeClasses` of the results of `HydrophilicityEvaluation()`

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Leslie A Kuhn, Craig A Swanson, Michael E Pique, John A Tainer, and Elizabeth D Getzof. Atomic and Residue Hydrophilicity in the Context of Folded Protein Structures. *PROTEINS: Structure, Function, and Genetics*, 1995, **23** (*4*), pp 536-547. DOI: 10.1002/prot.340230408 PMID: 8749849

See Also

Other "Hydrophilicity Evaluation" "Bound Water Environment": `HydrophilicityEvaluation`, `calcAtomHydrationEstimate`, `getProtAtomsNearWater`, `getResidueData`

Examples

```
 ## Not run:
  calcAtomClassHydrophilicity(df.AtomHydroTEMP)

## End(Not run)
```

`calcAtomHydrationEstimate`

Estimated Atomic Hydration Fraction

Description

Calculates the estimated atomic hydration fraction for an atom with unknown surface exposure.

Usage

```
calcAtomHydrationEstimate(df.AtomHydroTEMP, AT.hydratFract)
```

Arguments

df.AtomHydroTEMP	The newly calculated (determined) atomic hydrophilicity values
AT.hydratFract	The AtomTypeClasses.hydratFract variable calculated with the HydrophilicityEvaluation() function; the mean hydration fraction for the AtomTypes

Details

This function is called within HydrophilicityEvaluation() to calculate the estimated hydration of an atom with unknown surface exposure.

$$(numsurfaceexposures)/(numatomoccurrences) * atomclasshydrationfraction$$

NOTE: This is a non-public function.

Value

This function returns the hydration estimate values in a string to the variable AT.hydratFract.estimates and are included in the HydrophilicityTable results of HydrophilicityEvaluation().

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Leslie A Kuhn, Craig A Swanson, Michael E Pique, John A Tainer, and Elizabeth D Getzof. Atomic and Residue Hydrophilicity in the Context of Folded Protein Structures. *PROTEINS: Structure, Function, and Genetics*, 1995, **23** (*4*), pp 536-547. DOI: 10.1002/prot.340230408 PMID: 8749849

See Also

Other "Hydrophilicity Evaluation" "Bound Water Environment": HydrophilicityEvaluation, calcAtomClassHydrophilicity, getProtAtomsNearWater, getResidueData

Examples

```
 ## Not run:
  calcAtomHydrationEstimate(df.AtomHydroTEMP, AT.hydratFract)

## End(Not run)
```

`calcBvalue` *Calculate B-value*

Description

Calculate the B-value for an atom.

Usage

```
calcBvalue(rmsfValue)
```

Arguments

`rmsfValue`	rmsf value calculated by `bio3d::rmsf()`

Details

The B-value (aka B-factor) is calcualted from the rmsf from a collection of atoms. The rmsf is calculated using `bio3d::rmsf()`.

$$B - value = rmsf^2 * 8 * pi^2$$

The calculated B-values are returned within the `BoundWaterEnvironment()` results and used to define the size of conserved waters for the depiction of MDS conserved waters.

Value

B-value (aka B-factor) in Angstroms^2^

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Eaton E Lattman & Patrick J Loll. *Protein Crystallography: A Concise Guide*. Baltimore, Maryland, USA: The Johns Hopkins University Press, 2008. QP551.L345 2008. ISBN: 978-0-8018-8808-3 website

See Also

Other "Bound Water Environment": `BoundWaterEnvironment.interact`, `BoundWaterEnvironment.quality`, `BoundWaterEnvironment`, `Mobility`, `NormalizedBvalue`, `calcNearbyHydrationFraction`, `calcNumHydrogenBonds`

Examples

```
calcBvalue(rmsfValue=0.25)
# [1] 4.935
calcBvalue(rmsfValue=0.50)
# [1] 19.74
calcBvalue(rmsfValue=0.75)
# [1] 44.41
calcBvalue(rmsfValue=1.0)
# [1] 78.96
calcBvalue(rmsfValue=1.25)
# [1] 123.4
```

`calcNearbyHydrationFraction` *Calculate Nearby Atom Hydration Fraction*

Description

Calculate the mobility values of waters for a structure.

Usage

```
calcNearbyHydrationFraction(names.res.nearby.atoms)
```

Arguments

`names.res.nearby.atoms`
: string of residue-atom name for nearby atoms

Details

The summation, mean, and standard deviation of the hydrophilicity fraction for the protein atoms within the user specified distance for the `BoundWaterEnvironment()` function are calculated and returned.

Value

Hydrophilicity fraction sum, mean, and standard deviation.

Author(s)

Emilio Xavier Esposito `<emilio@exeResearch.com>`

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

See Also

Other "Bound Water Environment": BoundWaterEnvironment.interact, BoundWaterEnvironment.quality, BoundWaterEnvironment, Mobility, NormalizedBvalue, calcBvalue, calcNumHydrogenBonds

---

`calcNumHydrogenBonds` *Calculate Number of Hydrogen Bonds*

Description

Calculate the number of hydrogen bonds.

Usage

```
calcNumHydrogenBonds(distances, nearby.atoms.idc, names.atoms, set.oi.idc)
```

Arguments

`distances`	between water atom of interest and the protein atoms, water oxygen atoms, or HETATMs
`nearby.atoms.idc`	numeric vector of atom indices near water of interest
`names.atoms`	names of atoms; *e.g.*; c("CB", "CA", "N", "O", "CZ")
`set.oi.idc`	numeric vector of indices for protein atoms, water oxygen atoms, or HETATMs

Details

The summation, mean, and standard deviation of the hydrophilicity fraction for the protein atoms within the user specified distance for the `BoundWaterEnvironment()` function are calculated and returned.

Value

Number of possible hydrogen bonds between the water of interest and the protein atoms within 3.5 Angstroms of the water.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Bound Water Environment": BoundWaterEnvironment.interact, BoundWaterEnvironment.quality, BoundWaterEnvironment, Mobility, NormalizedBvalue, calcBvalue, calcNearbyHydrationFraction

Examples

```
  ## Not run:
  distances <- PDB.1hai.h2o.prot.dists[3, ]
  nearby.atoms.idc <- Nearby(distances, set.idc = prot.idc, radius = 3.6)
  names.atoms <- PDB.1hai.aoi.clean$elety[prot.idc]
  calcNumHydrogenBonds(distances, nearby.atoms.idc, names.atoms,
    set.oi.idc = prot.idc)
  # [1] 4

## End(Not run)
```

---

check.cluster.method *Check Clustering Method*

Description

Ensures the user provided clustering method is a valid choice.

Usage

```
check.cluster.method(cluster.method)
```

Arguments

`cluster.method` The user defined clustering method for the `ConservedWaters()` and `ConservedWaters.MDS()`.

Details

A simple check and reformatting of the clustering method indicated by the user in the `ConservedWaters()` and `ConservedWaters.MDS()` parameters.

Value

Correctly formatted clustering method or a stop error

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

CleanProteinStructures *Clean Protein Structures*

Description

Removes hydrogen and modeled atoms from a RCSB/PDB structure along with waters beyond a user defined distance from protein atoms.

Usage

```
CleanProteinStructures(prefix = "./alignTesting",
  CleanHydrogenAtoms = TRUE, CleanModeledAtoms = TRUE,
  cutoff.prot.h2o.dist = 6, min.num.h2o = 20,
  cleanDir = "ProteinSystem", filename = "ProteinSystem")
```

Arguments

`prefix`	The directory with the PDB files to be cleaned
`CleanHydrogenAtoms`	A logical indication if hydrogen atoms should be removed; default: `TRUE`
`CleanModeledAtoms`	A logical indication if modeled atoms should be removed; default: `TRUE`
`cutoff.prot.h2o.dist`	A numerical value setting the maximum distance between a protein atom (heteroatoms are ignored) and water oxygen atoms. The oxygen atoms equal to or less than this distance are retained; default: `6.0` Angstroms
`min.num.h2o`	Minimum number of water oxygen atoms within a protein structure for it to be included in the conserved water analysis; default: 20
`cleanDir`	A character string for the "cleaned" PDB structures to be written. The provided character string are appended with "_CLEANED"; default: `"ProteinSystem"`
`filename`	The filename prefix for the returned results. Default is "ProteinSystem"

Details

PDB files obtained from the PDB conform to a specific set of formatting standards but this does not mean the data within the PDB files is always correct. This function *cleans* the PDB file and summaries the atom evaluations.

This function does the following (in this order):

- Reads in the PDB file
- Adds/updates the element symbol (`elesy`) using the atom type (`elety`) via the `bio3d::atom2ele()` function
- Removes hydrogen atoms via `RemoveHydrogenAtoms()` (user option)
- Removes atoms with occupancy values determined to be out of range (OoR) via `RemoveOoR.o()`

- Removes atoms with B-values determined to be out of range (OoR) via `RemoveOoR.b()`
- Bins (counts) the occupancy values
- Bins (counts) the B-values
- Bins (counts) the normalized B-values
- Bins (counts) the mobility values
- Removes modeled atoms via `RemoveModeledAtoms()` (user option)
- Removes water oxygen atoms greater than user defined value `cutoff.prot.h2o.dist` from the protein via `RetainWatersWithinX()` (user option)
- Writes cleaned protein structure to a PDB file

Value

The following data is returned:

- **cleaning.summary**: summary indicating
  - if hydrogen atoms were removed `TRUE/FALSE`
  - number of out of range atoms for B-values and occupancy values
  - number of modeled (and thus removed)
  - number of atoms **NOT** modeled (and thus retained)
  - number of water oxygen atoms beyond the user defined cutoff
  - the number of water oxygen atoms within the user defined cutoff.
- **Bvalue.counts**: binned B-value values with binwidths = 5 (0 to 100)
- **normBvalue.counts**: binned normalized B-value values with binwidths = 0.1 (-4 to 6)
- **occupancy.counts**: binned occupancy values with binwidths = 0.1 (0 to 1)
- **mobility.counts**: binned mobility values with binwidths = 0.1 (0 to 6)
- **Excel workbook**: containing the cleaning.summary, Bvalue.counts, normBvalue.counts, occupancy.counts, and mobility.counts data as individual tabs
- **PDBids.retained**: a vector of PDBids
- **call**: parameters provided by the user

Author(s)

Emilio Xavier Esposito `<emilio@exeResearch.com>`

See Also

Other "Clean Protein Structure": `RemoveHydrogenAtoms`, `RemoveModeledAtoms`, `RemoveOoR.b`, `RemoveOoR.o`, `RetainWatersWithinX`

`ClusterSummaryPlots` *Cluster Summary Plots*

Description

Collection of cluster summary plots.

Usage

```
ClusterSummaryPlots(data, passed.waters = TRUE, plot.labels = NULL)
```

Arguments

`data`	The results from the `ConservedWaters()` function.
`passed.waters`	Logical indicator to plot results for waters **passing** `Mobility()` and `NormalizedBvalue()` **OR** using **all** waters within the `PDB` files.
`plot.labels`	Using the same options as `cowplot::plot_grid()` plus `NULL`. The option `"AUTO"` labels each plot with upper-case letters (*e.g.*, A, B, C, D, E), `"auto"` labels each plot with lower-case letters (*e.g.*, a, b, c, d, e), and `NULL` returns plots without labels. Default is `NULL`.

Details

The Number of Water Cluster (see `ConservationPlot()`), Occupancy (see `OccupancyBarplot()`), Mobility (see `MobilityBarplot()`), B-value (see `BvalueBarplot()`), and Normalized B-value (see `nBvalueBarplot()`) plots are combined into a single plot image. The ability to label each plot with capital letters (upper-case) or lower-case letters is available.

Author(s)

Emilio Xavier Esposito `<emilio@exeResearch.com>`

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

See Also

Other plots: `BoundWaterEnvPlots`, `BoundWaterEnvSummaryPlot`, `BvalueBarplot.summ`, `BvalueBarplot`, `MobNormBvalEvalPlots`, `MobilityBarplot.summ`, `MobilityBarplot`, `OccupancyBarplot.summ`, `OccupancyBarplot`, `nBvalueBarplot`, `normBvalueBarplot.summ`

Examples

```
## Not run:
cluster.summary.plot <- ClusterSummaryPlots(data=thrombin10.conservedWaters,
                                            passed.waters=TRUE,
                                            labels=NULL)

## End(Not run)
```

`ClusterWaters` *Cluster Conserved Waters*

Description

Cluster the conserved waters.

Usage

```
ClusterWaters(data, cutoff.cluster, cluster.method = "complete")
```

Arguments

`data`	The water oxygens' X, Y, and Z coordinates, B-values, and occupancy values.
`cutoff.cluster`	Numerical value provided by the user for the distance between water oxygen atoms to form a cluster; default: 2.4 Angstroms.
`cluster.method`	Method of clustering the waters; default is "`complete`". Any other method accepted by the `hclust` function is appropriate. The original method used by Sanschagrin and Kuhn is the complete linkage clustering method and is the default. Other options include "`ward.D`" (equivilant to the only Ward option in `R` versions 3.0.3 and earlier), "`ward.D2`" (implements Ward's 1963 criteria; see Murtagh and Legendre 2014), "`single`" (related to the minimal spanning tree method and adopts a "friend of friends" clustering method), along with "`average`" (= UPGMA), "`mcquitty`" (= WPGMA), "`median`" (= WPGMC) or "`centroid`" (= UPGMC). Due to size limitations with `stats::hclust()` – specifically the "`size cannot be NA nor exceed 65536`" – `fastcluster::hclust()` is being used because it is a complete replacement of `stats::hclust()`, is fast (compared to `stats::hclust()`), and is able to accommodate dissimilarity matrices with more than 2^16 (65,536) observations.

Details

Calculate the conserved waters using a collection of crystallographic protein structures.

Value

This function returns:

- **h2o.clusters.raw**: Initial waters with assigned cluster ID
- **h2o.clusters.summary**: Each cluster's:
  - cluster ID
  - number of waters
  - percent conservation
  - X, Y, and Z cooridinates
  - bound water environment measurements
  - mean distance between waters comprising the cluster
  - mean distance between waters comprising the cluster and the cluster's centroid
- **h2o.occurrence**: A table indicating the structures (PDBs) contributing to each cluster. This summary table includes the PDB structure's:
  - resolution
  - R-free value
  - occupancy (mean and standard deviation)
  - mobility (mean and standard deviation)
  - B-value (mean and standard deviation)
  - number of waters in each cluster
  - number of waters passing the mobility cutoff
  - number of waters passing the normalized B-value
  - number of waters passing both cutoff values
  - percentage of waters passing both cutoffs
  - number of clusters the structure contributes to
  - True/False table indicating if the protein structure contributed to the water cluster
- **clustering.info**: size and timing information

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064.
DOI: 10.1002/pro.5560071002
PMID: 9792092
WatCH webpage

Fionn Murtagh and Pierre Legendre. Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion? *Journal of Classification*, 2014, **31**, (*3*), pp 274-295.
DOI: 10.1007/s00357-014-9161-z

Daniel Müllner. fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *Journal of Statistical Software*, 2013, **53** (*9*)
DOI: 10.18637/jss.v053.i09 fastcluster webpage

`ClusterWaters.MDS` *Cluster Conserved Waters (MDS)*

Description

Cluster the conserved waters from a molecular dynamics simulation trajectory.

Usage

```
ClusterWaters.MDS(data, cutoff.cluster, cluster.method = "complete")
```

Arguments

`data`	The water oxygens' X, Y, and Z coordinates.
`cutoff.cluster`	Numerical value provided by the user for the distance between water oxygen atoms to form a cluster; default: 2.4 Angstroms.
`cluster.method`	Method of clustering the waters; default is "`complete`". Any other method accepted by the `hclust` function is appropriate. The original method used by Sanschagrin and Kuhn is the complete linkage clustering method and is the default. Other options include "`ward.D`" (equivilant to the only Ward option in `R` versions 3.0.3 and earlier), "`ward.D2`" (implements Ward's 1963 criteria; see Murtagh and Legendre 2014), "`single`" (related to the minimal spanning tree method and adopts a "friend of friends" clustering method), along with "`average`" (= UPGMA), "`mcquitty`" (= WPGMA), "`median`" (= WPGMC) or "`centroid`" (= UPGMC). Due to size limitations with `stats::hclust()` – specifically the "`size cannot be NA nor exceed 65536`" – `fastcluster::hclust()` is being used because it is a complete replacement of `stats::hclust()`, is fast (compared to `stats::hclust()`), and is able to accommodate dissimilarity matrices with more than 2^16 (65,536) observations.

Details

Calculate the conserved waters using a molecular dynamics simulation trajectory.

Value

This function returns:

- **h2o.clusters.raw**: Initial waters with assigned cluster ID
- **h2o.clusters.summary**: Each cluster's:
  - cluster ID
  - number of waters

  - percent conservation
  - X, Y, and Z cooridinates
  - bound water environment measurements
  - mean distance between waters comprising the cluster
  - mean distance between waters comprising the cluster and the cluster's centroid

- **h2o.occurrence**: A table indicating the structures (PDBs) contributing to each cluster. This summary table includes the PDB structure's:
  - number of waters in each cluster
  - number of clusters the structure contributes to
  - True/False table indicating if the protein structure contributed to the water cluster
- **clustering.info**: size and timing information

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064.
DOI: 10.1002/pro.5560071002
PMID: 9792092
WatCH webpage

Fionn Murtagh and Pierre Legendre. Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion? *Journal of Classification*, 2014, **31**, (*3*), pp 274-295.
DOI: 10.1007/s00357-014-9161-z

Daniel Müllner. fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *Journal of Statistical Software*, 2013, **53** (*9*)
DOI: 10.18637/jss.v053.i09 fastcluster webpage

`colorPalettes` *Color Values for Plots*

Description

Color values for plots with percent waters conserved plots.

Details

The five (5) and six (6) color palettes are to used to color-code the plots illustrating percent water conserved (water conservation). The five color palette is for conservation values between 50% to 100% and the six color palette includes a color for less than 50% conservation.

The colors are based on "percent conservation" with light grey dots indicating clusters with less than 50% conservation, dark red dots representing clusters with 50% to 69% conservations, red dots are clusters with 70% to 79% conservation, light blue dots have 80% to 89% conservation, blue dots are clusters with 90% to 99% conservation, and dark blue dots are 100% conserved water clusters (all structures contribute to the water cluster).

The defined colors are:

- **cons.color5**: red, medium red, light blue, medium blue, and dark blue
- **cons.color6**: light grey, red, medium red, light blue, medium blue, and dark blue

The defined legend titles are:

- **cons.color5.legend**: Water Conservation
- **cons.color6.legend**: Water Conservation

The defined break titles are:

- **cons.color5.breaks**: set1, set2, set3, set4, and set5
- **cons.color6.breaks**: set0, set1, set2, set3, set4, and set5

The defined labels are:

- **cons.color5.labels**: 50-69%, 70-79%, 80-89%, 90-99%, 100%
- **cons.color6.labels**: < 50%, 50-69%, 70-79%, 80-89%, 90-99%, 100%

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Cynthia A Brewer. 200x. http://www.ColorBrewer.org, accessed March 27, 2017.

`ConservationPlot` *Conservation Plot (Number of Waters Per Cluster Histogram)*

Description

Histogram and density plots for number of cluster with number of atoms

Usage

```
ConservationPlot(data, passed.waters = TRUE)
```

Arguments

data	The h2o.clusters.summary data.frame from the ClusterWaters() function containing the num.waters information. The num.waters values are integers.
passed.waters	Logical indicator to plot results for waters **passing** Mobility() and NormalizedBvalue() **OR** using **all** waters within the PDB files.

Details

Constructs a histogram for the number of waters per cluster. Clusters with less than 50% conservation are light grey, clusters with 50 to 69% water conseration are dark red, clusters with 70 to 79% conservation are red, 80 to 89% conservation are light blue, 90 to 99% conservation are blue, and 100% conservation (waters from all structures) are dark blue.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

Examples

```
## Not run:
Conservation.plot <- ConservationPlot(data=thrombin10.conservedWaters,
                                      passed.waters=TRUE)

## End(Not run)
```

ConservationSet *Conservation Set*

Description

Assign the percent conservation to a "set#" for plotting.

Usage

```
ConservationSet(pct.conserved)
```

Arguments

pct.conserved	A vector from containing the ConservedWaters() function containing the percent conservation (pct.conserved)

Details

Several of the plots color-code conserved water clusters based on percent conservation (see `ClusterSummaryPlots()` for color-coding) and is controlled by a `conserve.set` column. This function assigns less than 50% conservation to `set0`, 50 to 69% `set1`, 70 to 79% `set2`, 80 to 89% `set3`, 90 to 99% `set4`, and eqaul to 100% `set5`,

NOTE: This is a non-public function.

Value

vector indicating the conservation set

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: `DetermineChainsOfInterest`, `ExtractFileTimeStamp`, `ExtractPDBids`, `FileTimeStamp`, `HasXWaters`, `Nearby`, `ProtHetWatIndices`, `RescaleValues`, `RetainChainsOfInterest`, `ReturnPDBfullPath`, `StandardizeAsparticAcidNames`, `StandardizeCysteineNames`, `StandardizeGlutamicAcidNames`, `StandardizeHistidineNames`, `StandardizeLysineNames`, `TimeSpan`, `UniqueAtomHashes`, `aaStandardizeNames`, `getAtomTypeCounts`, `getResTypeCounts`, `res2xyz`, `resAtomType2AtomClass`, `write.basic.pdb`, `write.conservedWaters.pdb`

Examples

```
  ## Not run:
  pct.conserved <- c(100, 95, 90, 85, 80, 75, 70, 65, 60, 55, 50,
                     45, 40, 35, 30, 25, 20, 15, 10, 10)
  ConservationSet(pct.conserved)
  # [1] "set5" "set4" "set4" "set3" "set3" "set2" "set2" "set1" "set1" "set1"
  # "set1" "set0" "set0" "set0" "set0" "set0" "set0" "set0" "set0" "set0"

## End(Not run)
```

ConservedWaters *Conserved Crystallographic Waters*

Description

Identifies conserved crystallographic waters from a collection of PDBs.

Usage

```
ConservedWaters(prefix = "", cluster = 2.4, mobility = 2,
  nBvalue = 1, chain = "first", prot.h2o.dist.min = 5.1,
  cluster.method = "complete", PDBinfo, filename = "ProteinSystem")
```

Arguments

prefix	Directory of aligned structures; string.
cluster	Oxygen atoms within 2.4 Angstroms or less of each other are considered a cluster; numeric. Default value is 2.4 Angstroms.
mobility	A normalization method to identify the amount of variance an atom has within a structure; numeric. Calculated mobility values equal to or greater than the provided value will be removed from analysis. Default value is 2.0. See `Mobility()` for more information.
nBvalue	The number of standard deviations from the mean for the water oxygens' B-values within the structure of interest; numeric. Calculated normalized B-values equal to or greater than the provided value will be removed from analysis. Default value is 1.0. See `NormalizedBvalue()` for more information.
chain	The chain to examine. The user can define "first" and the first chain alphabetically will be selected; this is the default. Defining "all" will result in all chains being explored. Alternatively the user can define individual the chains to include in the analysis; for example, `c("A", "B", "C")`. When defining chains, the chain designation **must be characters**.
prot.h2o.dist.min	The minimum distance (in Angstroms) between the protein and waters to be considered for the conserved water clusters. Water oxygen atoms greater than this distance are removed from the analysis. Default value is 5.10 Angstroms.
cluster.method	Method of clustering the waters; default is "complete". Any other method accepted by the `stats::hclust()` or `fastcluster::hclust()` functions are appropriate. The original method used by Sanschagrin and Kuhn is the complete linkage clustering method and is the default. Other options include "ward.D" (equivilant to the only Ward option in R versions 3.0.3 and earlier), "ward.D2" (implements Ward's 1963 criteria; see Murtagh and Legendre 2014), or "single" (related to the minimal spanning tree method and adopts a "friend of friends" clustering method). Please see `fastcluster::hclust()` for additional and complete information regarding clustering explanations.
PDBinfo	The PDB information for all structures within the analysis. This information is obtained using the `getRCSBdata()` function.
filename	The filename prefix for the returned results. Default is "ProteinSystem"

Details

Only atoms within (less than or equal to) 5.10 Angstroms of the protein structures are included.

Value

This function returns:

- **h2o.cluster.all**: Clusters constructed from all waters present in the aligned PDB structures.
- **h2o.cluster.passed**: Clusters constructed from waters that passed the `Mobility()` and `NormalizedBvalue()` evaluations.
- **h2o.cluster.summary**: Summary of water clusters

- **Excel workbook**: containing the Cluster Statistics, Cluster Summaries for **all** and **passed** waters, Occurrence Summaries for **all** and **passed** waters, and the Initial Water Data data as individual tabs
- **call**: The user provided parameters for the function

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

Hitesh Patel, Bjorn A. Gruning, Stefan Gunther, and Irmgard Merfort. PyWATER: a PyMOL plug-in to find conserved water molecules in proteins by clustering. *Bioinformatics*, 2014, **30** (*20*), pp 2978-2980. DOI: 10.1093/bioinformatics/btu424 PMID: 24990608 PyWATER on GitHub

---

ConservedWaters.MDS *Conserved Molecular Dynamics Simulation Waters*

Description

Identifies conserved molecular dynamics simulation (MDS) waters from a collection of PDBs.

Usage

```
ConservedWaters.MDS(prefix = "", cluster = 2.4, chain = "all",
  prot.h2o.dist.min = 5.1, cluster.method = "complete",
  filename = "ProteinSystem")
```

Arguments

`prefix`	Directory of aligned structures; string.
`cluster`	Oxygen atoms within 2.4 Angstroms or less of each other are considered a cluster; numeric. Default value is 2.4 Angstroms.
`chain`	The chain to examine. The user can define "first" and the first chain alphabetically will be selected; this is the default. Defining "all" will result in all chains being explored. Alternatively the user can define individual the chains to include in the analysis; for example, `c("A", "B", "C")`. When defining chains, the chain designation **must be characters**.
`prot.h2o.dist.min`	The minimum distance (in Angstroms) between the protein and waters to be considered for the conserved water clusters. Water oxygen atoms greater than this distance are removed from the analysis. Default value is 5.10 Angstroms.

`cluster.method` Method of clustering the waters; default is "complete". Any other method accepted by the `stats::hclust()` or `fastcluster::hclust()` functions are appropriate. The original method used by Sanschagrin and Kuhn is the complete linkage clustering method and is the default. Other options include "ward.D" (equivilant to the only Ward option in `R` versions 3.0.3 and earlier), "ward.D2" (implements Ward's 1963 criteria; see Murtagh and Legendre 2014), or "single" (related to the minimal spanning tree method and adopts a "friend of friends" clustering method). Please see `fastcluster::hclust()` for additional and complete information regarding clustering explanations.

`filename` The filename prefix for the returned results. Default is "ProteinSystem"

Details

Only atoms within (less than or equal to) 5.10 Angstroms of the protein structures are included.

Value

This function returns:

- **h2o.cluster.all**: Clusters constructed from all waters present in the aligned PDB structures.
- **h2o.cluster.passed**: Clusters constructed from waters that passed the `Mobility()` and `NormalizedBvalue()` evaluations.
- **h2o.cluster.summary**: Summary of water clusters
- **Excel workbook**: containing the Cluster Statistics, Cluster Summaries for **all** and **passed** waters, Occurrence Summaries for **all** and **passed** waters, and the Initial Water Data data as individual tabs
- **call**: The user provided parameters for the function

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

Hitesh Patel, Bjorn A. Gruning, Stefan Gunther, and Irmgard Merfort. PyWATER: a PyMOL plug-in to find conserved water molecules in proteins by clustering. *Bioinformatics*, 2014, **30** (*20*), pp 2978-2980. DOI: 10.1093/bioinformatics/btu424 PMID: 24990608 PyWATER on GitHub

`ConservedWaterStats` *Conserved Water Statistics*

Description

Calculates the Conserved Water Statistics for `ConservedWaters()`

Usage

```
ConservedWaterStats(h2o.cluster, num.h2o.inital, num.pdbs.got.h2o)
```

Arguments

`h2o.cluster`	Conserved water cluster
`num.h2o.inital`	Number of initial waters
`num.pdbs.got.h2o`	Number of PDB structures with waters

Details

Calculates the statistics for each conserved water analysis performed by `ConservedWaters()`. This summary information is useful for timings information and is written to the Excel workbook.

Value

A table with the following information is returned:

- Number of structures
- Number of initial waters
- Number of waters used in the calculations
- Number of water clusters
- Average water conservation
- Number of conserved waters with
  - < 50% conservation
  - 50 - 69% conservation
  - 70 - 79% conservation
  - 80 - 89% conservation
  - 90 - 99% conservation
  - 100% conservation
- Number of pairwise distances evaluated
- Amount of memory used by the pairwise diatance matrix
- Pairwise distance calculation time
- Cluster centroid distance calculation time

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "vanddraabe utilities": FreeSASAcheck

`CreatePyMOLscript` *Create PyMOL Script File*

Description

Create PyMOL script file to visualize conserved waters

Usage

```
CreatePyMOLscript(conservedWaters.data, passed.waters = TRUE,
  PDBid.ref = "1hai", LigResname.ref = NULL, hbond = 3.75,
  lig.carbon.color = "cyan", filename = "thrombin10")
```

Arguments

`conservedWaters.data`	The `h2o.clusters.summary` data.frame from the `ConservedWaters()` function containing the `nBvalue.mu` information. This data.frame is found within the `h2o.cluster.passed` and `h2o.cluster.all`
`passed.waters`	Logical indicator to plot results for waters **passing** `Mobility()` and `NormalizedBvalue()` **OR** using **all** waters within the `PDB` files.
`PDBid.ref`	name for reference structure in PyMOL; *e.g.*, `"1hai"`
`LigResname.ref`	PDB residue code for reference ligand; *e.g.*, `"0g6"`
`hbond`	The minimum distance between hydrogen bond acceptor and donor; default: `3.75`
`lig.carbon.color`	One of the ten pre-defined carbon-color options using PyMOL's `util.cbaX` command. The `X` represents the user defined color of carbon atoms. `X` can be `g`: green; `c`: cyan; `m`: magenta; `y`: yellow; `s`: salmon; `w`: grey; `b`: slate; `o`: orange; `p`: purple; and `k`: pink; default: `cyan`
`filename`	Prefix for the PyMOL script files. It is probably best to use the initial portion of the conserved waters PDB filename; *e.g.*, `"thrombin10"`

Details

The ability to visualize the conserved waters is important and their surroundings is when exploring conserved water results.

Conserved waters within 6 Angstroms of the PyMOL identified ligands are displayed. The conserved waters are colored based on their percent conservation range using the same color scheme as the Percent Conservation plot. Waters conserved less than 50% are colored light grey, 50-69% are red, 70-79% are dark red, 80-89% are light blue, 90-99% are medium blue, and 100% are dark blue. The conserved waters are labeled using their ranking based on percent conseration.

This function creates **two** PyMOL script files; one with a black background and another with a white background. The color of the pocket residues is changed based on the background. The pocket residues are colored light-grey for the black background and dark-grey for the white background. The ligand is assigned the user-defined color for both representations. Pocket residues – and associated molecular surface – are defined as those within 5 Angstroms of the conserved waters. The depicted cartoon representation is for residues within 15 Angstroms of the ligand(s).

The potential hydrogen bonds are depicted between:

- conserved waters and ligand: orange dashed line
- conserved waters and protein: green dashed line
- conserved waters: blue dashed line

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

Examples

```
## Not run:
current.time <- Sys.time()
CreatePyMOLfile(PDBid.ref = "Thrombin_initial10_alignedGood/1hai_aligned_pruned.pdb",
                PDBid.ref = "1hai",
                LigResname.ref = "0g6",
          conserved.waters = "Thrombin_initial10_ConservedWaters_PASSED_mar292017_1535.pdb",
                hbond = 3.75,
                lig.carbon.color = "cyan",
                filename = "thrombin10_ConservedWaters_PASSED")

## End(Not run)
```

DetermineChainsOfInterest

Determine Chains Of Interest

Description

Determine the chains identification

Usage

```
DetermineChainsOfInterest(chains.to.explore)
```

Arguments

chains.to.explore	**NOTE**: "first" is alphabetically first. Thus if the order within the original PDB file is L and then H, this function will return H because it is alphebetically first.

Details

Standardizes user provided chain(s) of interest. This function simply standardizes the user provided chains of interest. Acceptable values are: - **first**: alphabetically the first chain - **all**: all chains within a structure file - **user defined**: a single letter or a set of letters; *e.g.*; "A" or c("H", "L")

NOTE: This is a **non-public** function and is **NOT** available for general use. Please contact the author if you believe this function should be available for general use.

Value

string indicating which chain designation (*e.g.*, "first" chain, "all" chains, or "user" defined) to include in the conserved water analysis

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
## Not run:
DetermineChainsOfInterest("first")
# [1] "first"
DetermineChainsOfInterest("ALL")
# [1] "all"
DetermineChainsOfInterest("D")
# [1] "user"
DetermineChainsOfInterest(c("H", "L"))
# [1] "user"
DetermineChainsOfInterest("vanddraabe")
# The provided chain ID VANDDRAABE is not valid and the first chain will
# be used; likely chain A.
```

```
  # [1] "first"

## End(Not run)
```

`ExtractFileTimeStamp` *Extract Filename Time Stamp*

Description

Extract date & time stamp from a file

Usage

```
ExtractFileTimeStamp(filename)
```

Arguments

`filename`	String of the file name to extract the **FileTimeStamp** information

Details

Create a date-time string to append to filenames to try and make them unique. The date-time string has the format month-day-year_hour-minute; for example, May 4, 2016 at 12:34pm is represented as `may042016_1234`.

NOTE: This is a non-public function.

Value

A string with the date and time.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: `ConservationSet`, `DetermineChainsOfInterest`, `ExtractPDBids`, `FileTimeStamp`, `HasXWaters`, `Nearby`, `ProtHetWatIndices`, `RescaleValues`, `RetainChainsOfInterest`, `ReturnPDBfullPath`, `StandardizeAsparticAcidNames`, `StandardizeCysteineNames`, `StandardizeGlutamicAcidNames`, `StandardizeHistidineNames`, `StandardizeLysineNames`, `TimeSpan`, `UniqueAtomHashes`, `aaStandardizeNames`, `getAtomTypeCounts`, `getResTypeCounts`, `res2xyz`, `resAtomType2AtomClass`, `write.basic.pdb`, `write.conservedWaters.pdb`

Examples

```
## Not run:
filename <- "ConservedWaters_PASSED_may042016_1234.pdb"
ExtractFileTimeStamp(filename)
# [1] may042016_1234

## End(Not run)
```

`ExtractPDBids` *Extract PDB IDs*

Description

Extract the four (4) character PDB identifier from the file name

Usage

```
ExtractPDBids(pdb.location)
```

Arguments

`pdb.location`	A collection of string values with the complete (normalized) path for each PDB file within the provided directory/folder obtained with the `ReturnPDBfullPath()`.

Details

The first four (4) characters of the file name – typically the PDB ID is placed at the beginning of the file name – are extracted and assumed to be the unique PDB ID.

NOTE: This is a non-public function.

Value

a vector of strings containing the PDB identifiers for the protein structures

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: `ConservationSet`, `DetermineChainsOfInterest`, `ExtractFileTimeStamp`, `FileTimeStamp`, `HasXWaters`, `Nearby`, `ProtHetWatIndices`, `RescaleValues`, `RetainChainsOfInterest`, `ReturnPDBfullPath`, `StandardizeAsparticAcidNames`, `StandardizeCysteineNames`, `StandardizeGlutamicAcidNames`, `StandardizeHistidineNames`, `StandardizeLysineNames`, `TimeSpan`, `UniqueAtomHashes`, `aaStandardizeNames`, `getAtomTypeCounts`, `getResTypeCounts`, `res2xyz`, `resAtomType2AtomClass`, `write.basic.pdb`, `write.conservedWaters.pdb`

Examples

```
  ## Not run:
  ExtractPDBids("1hai.pdb")
  # [1] "1hai"
  ExtractPDBids("/home/someuser/pdbs/1hai.pdb")
  # [1] "1hai"

## End(Not run)
```

`FileTimeStamp` *Filename Time Stamp*

Description

Date-time string to make file names unique

Usage

```
FileTimeStamp(current.time)
```

Arguments

`current.time`	The current time determined with `base::as.POSIXct()`

Details

Create a date-time string to append to filenames to try and make them unique. The date-time string has the format month-day-year_hour-minute; for example, May 4, 2016 at 12:34pm is represented as `may042016_1234`.

NOTE: This is a non-public function.

Value

A string with the date and time.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: `ConservationSet`, `DetermineChainsOfInterest`, `ExtractFileTimeStamp`, `ExtractPDBids`, `HasXWaters`, `Nearby`, `ProtHetWatIndices`, `RescaleValues`, `RetainChainsOfInterest`, `ReturnPDBfullPath`, `StandardizeAsparticAcidNames`, `StandardizeCysteineNames`, `StandardizeGlutamicAcidNames`, `StandardizeHistidineNames`, `StandardizeLysineNames`, `TimeSpan`, `UniqueAtomHashes`, `aaStandardizeNames`, `getAtomTypeCounts`, `getResTypeCounts`, `res2xyz`, `resAtomType2AtomClass`, `write.basic.pdb`, `write.conservedWaters.pdb`

Examples

```
## Not run:
current.time <- as.POSIXct("2016-05-04 12:34:56.78", tz = "UTC")
FileTimeStamp(current.time)
# [1] may042016_1234

## End(Not run)
```

`FreeSASA.diff` *Atomic SASA difference of hydrated PDB via FreeSASA*

Description

Calculates the atomic solvent accessible surface area (SASA) of the provided PDB (protein structure) using the FreeSASA application (website).

Usage

```
FreeSASA.diff(atoms.oi, probeRadius = 1.4)
```

Arguments

`atoms.oi`	PDB structure read into R by `bio3d::read.pdb()`; the `base::data.frame()` of `pdb$atom`
`probeRadius`	Numerical values indicating the probe radius in Angstroms for the FreeSASA application; default: 1.4

Details

The purpose of this function is to calculate and return the calculated atomic SASA for the provided PDB (protein structure) and the SASA of the protein when including the hydrating waters.

Several of the FreeSASA options are set and **NOT** user changeable. Specifically, no log information is returned; the `-L` ; the number of slices per atom is set to the FreeSASA default of 20 (Lee & Richards algorithm); each FreeSASA calculation uses four (4) threads; and the ProtOr atomic radii are used.

It might be too late if you are reading this, but it is **strongly** encouraged to run `FreeSASAcheck()` to check if the FreeSASA application is correctly installed.

Value

A PDB list with FreeSASA (ProtOr) atomic radii placed in the *occupancy (o)* column and SASA values calculated using the Lee & Richards method in the *b-value (b)* column.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

ProtOr (Protein-Organic) atomic radii:
Jerry Tsai, Robin Taylor, Cyrus Chothia, and Mark Gerstein. The packing density in proteins: standard radii and volumes. *J Mol Biol*, 1999, **290** (*1*), pp 253-266. DOI: 10.1006/jmbi.1999.2829 PMID: 10388571

SASA calculation method:
B Lee, FM Richards. The interpretation of protein structures: estimation of static accessibility. *J Mol Biol*, 1971, **55** (*3*), pp 379-400. DOI: 10.1016/0022-2836(71)90324-X

FreeSASA application:
Simon Mitternacht. FreeSASA: An open source C library for solvent accessible surface area calculations [version 1; referees: 2 approved]. *F1000Research*, 2016, **5**:189 DOI: 10.12688/f1000research.7931.1 PMCID: PMC4776673 FreeSASA

Examples

```
## Not run:
  SASA.diff <- FreeSASA.diff(atoms.oi = thrombin.1hai$atom,
                             probeRadius = 1.4)
  head(SASA.diff)
  #   uniq.atom.ids SASA.prot SASA.hetatm SASA.lost
  # 1   THR_1_L_N_1      0.00        0.00         0
  # 2  THR_1_L_CA_2      0.00        0.00         0
  # 3   THR_1_L_C_3      0.00        0.00         0
  # 4   THR_1_L_O_4      0.00        0.00         0
  # 5  THR_1_L_CB_5      1.92        1.92         0
  # 6 THR_1_L_OG1_6     11.25       11.25         0
  #
  stem(SASA.diff$SASA.lost)
  #
  # The decimal point is at the |
  #
  #  0 | 000000000000000000000000000000000000000000000000000000000000000000000000+1721
  #  2 | 000000000000011111111111222222222233333333334444444455555555555566666+88
  #  4 | 000011111111222223333333333444444445555556667777777778889999000000+23
  #  6 | 000001111112222222222333344445555666667777888990000111222233333445 5
  #  8 | 00112222233334456677788899001111222223444556667899
  # 10 | 000000001233445567888990001333344567899999
  # 12 | 000012333344466788001112333447888 89
  # 14 | 00134448881223589
  # 16 | 014466389
  # 18 | 0945578888
  # 20 | 22347702
  # 22 | 246999
  # 24 | 09457
  # 26 | 44
  # 28 | 6
  # 30 |
  # 32 |
  # 34 | 9
  # 36 |
  # 38 |
```

```
  # 40 |
  # 42 |
  # 44 |
  # 46 | 9
  #

## End(Not run)
```

FreeSASAcheck *FreeSASA Check*

Description

Determines if FreeSASA is (correctly) installed.

Usage

```
FreeSASAcheck()
```

Details

Because FreeSASA is **NOT** included with vanddraabe it is important to ensure the application has been installed and was correctly compiled.

Value

When FreeSASA is correctly installed the current version and citation are returned to the user:

```
FreeSASA 2.0
License: MIT <http://opensource.org/licenses/MIT>
If you use this program for research, please cite:
  Simon Mitternacht (2016) FreeSASA: An open source C
  library for solvent accessible surface area calculations.
F1000Research 5:189.
```

When FreeSASA is **NOT** correctly installed the following are returned to the user:

```
Error in FreeSASAcheck() :
Uh-oh!!
Please make sure FreeSASA is correctly installed! Please visit
(http://freesasa.github.io) for
instructions specific to your operating system.
```

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Simon Mitternacht. FreeSASA: An open source C library for solvent accessible surface area calculations [version 1; referees: 2 approved]. *F1000Research*, 2016, **5**:189 DOI: 10.12688/f1000research.7931.1 PMCID: PMC4776673 FreeSASA

See Also

Other "vanddraabe utilities": `ConservedWaterStats`

Examples

```
## Not run:
  # Result for correct installation
  FreeSASAcheck()
  # FreeSASA 2.0
  # License: MIT <http://opensource.org/licenses/MIT>
  # If you use this program for research, please cite:
  #   Simon Mitternacht (2016) FreeSASA: An open source C
  #   library for solvent accessible surface area calculations.
  #   F1000Research 5:189.
  #
  # Report bugs to <https://github.com/mittinatten/freesasa/issues>
  # Home page: <http://freesasa.github.io>
  #
  # Congratulations! FreeSASA is correctly installed!
  #
  # Result for incorrect installation
  FreeSASAcheck()
  # Error:
  # Uh-oh!!
  # Please make sure FreeSASA is correctly installed. Please visit
  # http://freesasa.github.io for instructions specific to your operating
  # system.

## End(Not run)
```

`getAtomTypeCounts` *Get AtomType Counts*

Description

Counts the number of AtomTypes within the provided string.

Usage

```
getAtomTypeCounts(atom.types)
```

Arguments

atom.types	A vector of strings containing a combination of the 167 AtomTypes.

Details

This is a wrapper using the base::table() function. The vector of AtomTypes (strings) are passed to the function, non-standard AtomTypes are removed, the AtomTypes are counted, and the counts are ordered based on the names.AtomTypes constant.

NOTE: This is a non-public function.

Value

a vector of numbers indicating the counts of each AtomType. The vector is ordered based on the names.AtomTypes with AtomTypes not included assigned a value of zero (0).

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
## Not run:
set.seed(13)
num.AtomTypes <- sample(1:10, 30, replace = TRUE)
atom.types <- rep(sample(names.res.AtomTypes, 30), num.AtomTypes)
getAtomTypeCounts(atom.types)
# [1]  0  0  0  0  0  0  0  0  0  0  0  0  0  7  1  0  0  0  0
#      0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
#      0  0  0  4  0  0  0  0  4  3  6  0  0  0  0  0  0  0  0
#      0  0 10  0  8  0  0  0  0  0  0  0  0  0  0  0  0  0  6
#      4  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#      0  5  0  0  0  0  0  0  0  7  0  0  0  0  0  0  0  7  0
#      3  0  0  5  3  0  5  0  0  5  8  0  8  0  8  0  0  0  0
#      0  0  2  0  0  1  0  0  0  3  0  0  6  0  0  0  0  0  3
#      0  0  0  0 10  0  0  0  0  0  0  1  0  0  0

## End(Not run)
```

`getProtAtomsNearWater` *Number of Solvent Accessible/Exposed Protein Atoms Near a Water*

Description

Calculate the number of solvent exposed protein atoms near a water.

Usage

```
getProtAtomsNearWater(h2o.oi, h2o.idc, atoms.oi, h2o.prot.dists,
  h2o.prot.dists.tf)
```

Arguments

`h2o.oi`	The index of the water of interest
`h2o.idc`	The indices of the waters within the protein structure
`atoms.oi`	The protein `data.frame` with the `SASA` and `SASA lost` values for each atom within the protein.
`h2o.prot.dists`	Distance `matrix` for all water-protein through space distances
`h2o.prot.dists.tf`	The `TRUE/FALSE` matrix indicating if the protein- water distances are less than or equal to the user defined cutoff value denoted by the `h2o.prot.dist.max` parameter for `HydrophilicityEvaluation()`. From `HydrophilicityEvaluation()`: the maximum distance between the water oxygen atoms and the protein for consideration in the determination for hydrophilicity values; default: 6.0

Details

This function is called within `HydrophilicityEvaluation()` to determine protein atoms near each water oxygen.

This function is designed to work with the `base::lapply()` function and thus each `h2o.oi` is independently evaluated

Value

This function returns a `data.frame` with:

- **nearby.prot.atoms**: protein atoms within the user specified distance of a water's oxygen atom
- **distances**: The distance – in Angstroms – from the water to the closest solvent accessible protein atom so long as the distance is equal to or less than the user provided value; see `h2o.prot.dists.tf` above
- **dist.is.min**: ; see `h2o.prot.dists.tf` above
- **SASA.and.minDist**: `TRUE/FALSE` indicating if the protein atom is **BOTH** solvent accessible and at least the user defined number of Angstroms from a water's oxygen atom; see `h2o.prot.dists.tf` above

- **h2o.atom.ids**: Unique water atom ID
- **h2o.x**: Atom coordinate `X` for the water's oxygen atom
- **h2o.y**: Atom coordinate `Y` for the water's oxygen atom
- **h2o.z**: Atom coordinate `Z` for the water's oxygen atom

These values are returned in `df.nearby.prot.atoms` of the results of `HydrophilicityEvaluation()`

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Hydrophilicity Evaluation" "Bound Water Environment": `HydrophilicityEvaluation`, `calcAtomClassHydrophilicity`, `calcAtomHydrationEstimate`, `getResidueData`

Examples

```
  ## Not run:
  getProtAtomsNearWater(h2o.oi = PDB.1hai.h2o.oi,
                        h2o.idc = PDB.1hai.clean.h2o.idc,
                        atoms.oi = PDB.1hai.aoi.clean.SASA,
                        h2o.prot.dists = PDB.1hai.h2o.prot.dists,
                        h2o.prot.dists.tf = PDB.1hai.h2o.prot.dists.tf)

## End(Not run)
```

---

getRCSBdata *Clean RCSB Dataset*

Description

Clean the protein dataset based on quality values.

Usage

```
getRCSBdata(prefix = "./alignTesting", resolution = 3, rFree = 0.26,
  rObserved = 0.2, filename = "ProteinSystem")
```

Arguments

`prefix`	Directory of aligned structures; string.
`resolution`	Structures with a `resolution` value greater than this value are removed from analysis; default: `3.0`
`rFree`	Structures with a `rFree` values greater than this value are removed from analysis; default: `0.26`

rObserved	Structures with a rObserved values greater than this value are removed from analysis; default: 0.20
filename	The filename prefix for the returned results. Default is "ProteinSystem"

Details

The provided protein models determined by X-ray crystallography and downloaded from the RCSB include structure quality measures. The resolution, rObservation, and rFree are the three commonly used and referenced evaluation measures.

The B-value normalization exclusion value is user defined within the main ConservedWaters() function but has a default value of 1.0.

Value

This function returns:

- **PDB.info**: RCSB provided information for all protein structures
- **PDB.info.passed**: RCSB provided information for all protein structures **passing** the user defined parameters
- **PDB.info.rejected**: RCSB provided information for all protein structures **failing** the user defined parameters
- **call**: parameters provided by the user
- **Excel workbook**: containing the PDB.info, PDB.info.passed, and PDB.info.rejected data as individual tabs

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

Examples

```
## Not run:
proteins.info <- getRCSBdata(prefix="./thrombin_fitlsq_1.0ang/",
                             resolution=3.0, rFree=NULL, rObserved= 0.20,
                             filename="ProteinSystem")

## End(Not run)
```

---

getResidueData *Number of Residues and Solvent Accessible/Exposed Residues*

Description

Calculate the number of residues and solvent exposed residues.

Usage

```
getResidueData(atoms.oi.prot, SurExp.res.atoms.tf)
```

Arguments

atoms.oi.prot	The protein data.frame with the SASA and SASA lost values for each protein atom.
SurExp.res.atoms.tf	TRUE/FALSE vector indicating if an atom is solvent exposed/accessible

Details

This function is called within HydrophilicityEvaluation() to provide general solvent accessibility data for the protein structure of interest.

Value

This function returns:

- **num.res**: number of residues within the structure
- **num.res.buried**: number of residues with **NO** solvent accessible surface area
- **num.res.SurExp**: number of residues with solvent accessible surface area
- **pct.res.SurExp**: percentage of residues with solvent accessible surface area
- **SASA.total**: total protein solvent accessible surface area; Angstroms^2^
- **SASA.lost**: total protein solvent accessible surface area lost due to bound waters; Angstroms^2^
- **pct.SASA.exposed**: percentage protein solvent accessible surface area $(SASA.total - SASA.lost)/SASA.total$

These values are returned in df.residue.hydro of the results of HydrophilicityEvaluation()

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Hydrophilicity Evaluation" "Bound Water Environment": HydrophilicityEvaluation, calcAtomClassHydrophilicity, calcAtomHydrationEstimate, getProtAtomsNearWater

Examples

```
  ## Not run:
  getResidueData(atoms.oi.prot = PDB.1hai.aoi.clean.SASA.prot,
        SurExp.res.atoms.tf = PDB.1hai.SurExp.res.atoms.tf)

## End(Not run)
```

getResTypeCounts *Get ResType Counts*

Description

Counts the number of ResType within the provided string.

Usage

```
getResTypeCounts(res.types)
```

Arguments

res.types	A vector of strings containing a combination of the 20 ResTypes.

Details

This is a wrapper for the base::table() function. The vector of ResType are passed to the function, non-standard ResType are removed, the ResType are counted, and the counts are ordered based on the names.ResTypes constant.

NOTE: This is a non-public function.

Value

a vector of numbers indicating the counts of each ResType. The vector is ordered based on the names.ResTypes with ResTypes not included assigned a value of zero (0).

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
## Not run:
set.seed(13)
num.ResTypes <- sample(1:10, 20, replace = TRUE)
res.types <- rep(names.residues, num.ResTypes)
getResTypeCounts(res.types)
# [1] 8  3  5 10  6  6  4  8  3  1 10  7  7  5  1  6  8  1  1  4
```

```
## End(Not run)
```

HasXWaters *Has "X" Waters*

Description

Determines if PDB structure has water molecules.

Usage

```
HasXWaters(atoms.oi.resid, min.num.h2o = 20)
```

Arguments

atoms.oi.resid	vector of character strings containing the standardized three-letter amino acid residue names
min.num.h2o	numeric value indicating the minimum number of water molecules required to return a TRUE logical value

Details

Determine if the PDB structure has at least the user defined number of water oxygen atoms. The number of water oxygen atoms is returned along with a logical value indicating if the structure satisfies the user defined minimum.

Waters are identified using the three water three-letter residue names: HOH, WAT, and DOD.

Value

logical indicating if the PDB structure has the minimum user defined number of waters

numeric value indicating the number of water oxygen atoms within the PDB structure

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
resids <- c("ALA", "HOH", "WAT", "ALA", "HOH", "DOD", "ALA", "HOH")
HasXWaters(resids, min.num.h2o = 4)
# $has.h2o.tf
# [1] TRUE
#
# $num.water
# [1] 5
```

`HydrophilicityEvaluation`

Hydrophilicity Evaluation

Description

Calculate the hydrophilicity values for a set of protein structures.

Usage

```
HydrophilicityEvaluation(prefix = "alignTesting/",
  h2o.prot.dist.max = 6, bound.h2o.dist.max = 4, min.num.h2o = 20,
  probeRadius = 1.4, dataset = "top56")
```

Arguments

`prefix`	The directory containing the protein structures; *e.g.*, "alignTesting/"
`h2o.prot.dist.max`	Maximum distance between the water oxygen atoms and the protein for consideration in the determination for hydrophilicity values; default: 6.0
`bound.h2o.dist.max`	Maximum distance between the water oxygen atoms and the protein for inclusion in the calculation of hydrophilicity values; default: 4.0
`min.num.h2o`	Minimum number of water oxygen atoms within a protein structure for it to be included in the calculation of hydrophilicity values; default: 20
`probeRadius`	Water molecule probe radius; default: 1.4
`dataset`	Name of the dataset to be used; *e.g.*,"top56"

Details

The hydrophilicity values of individual atomtypes is determined using a collection of protein structures. For each water oxygen atom within at the most 4 Angstroms of a solvent accessible (exposed) protein atom, these occurrences are recorded. The number of solvent accessible atom types interacting with a water molecule are divided by the number of solvent accessible atom types. In general the more diverse data available, the better the informatics based hydrophilicity values should correlate with various experimental values.

NOTE: Hydrogen atoms are removed for instances when the protein structures have not be cleaned with `CleanProteinStructures()`.

Value

This function returns:

- **PDB.info**: a summary of the data for each protein structure analyzed
  - *PDBid*: PDB id
  - *time*: duration for hydrophilicity evaluation
  - *num.res*: number of protein residues
  - *num.res.buried*: number of protein residues with **NO** solvent exposure
  - *num.res.SurExp*: number of protein residues with solvent accessible surface area
  - *pct.res.SurExp*: percentage of protein residues with solvent
  - *SASA.total*: total protein solvent accessible surface area; Angstroms^2^
  - *SASA.lost*: total protein solvent accessible surface area lost due to bound waters; Angstroms^2^
  - *pct.SASA.exposed*: percentage protein solvent accessible surface area $(SASA.total - SASA.lost)/SASA.total$
  - *num.prot.atom*: number of protein atoms
  - *num.atom.buried*: number of protein atoms with **NO** solvent exposure
  - *num.atom.SurExp*: number of protein atoms with solvent accessible surface area
  - *pct.atom.SurExp*: percentage protein atoms with solvent accessible surface area $(SASA.total - SASA.lost)/SASA.total$
  - *num.h2o*: number of waters in the system
  - *num.h2o.lte.prot.max*: number of waters within `h2o.prot.dist.max` cutoff
  - *num.SurBound.h2o*: number of surface bound waters; water within `bound.h2o.dist.max` cutoff
  - *num.bb.h2o.inter*: number of backbone - water interactions
  - *num.sc.h2o.inter*: number of sidechain - water interactions
  - *num.res.h2o.inter*: number of interactions between residues and water
  - *num.h2o.res.inter*: number of interactions between water and residue (residues are a unit)
  - *num.h2o.resAtom.inter*: number of water-atom interactions
- **SASA.results**: `data.frame` of protein atoms within the `h2o.prot.dist.max` of each water oxygen atom
- **df.AtomTypes.all**: total number of AtomTypes for each structure
- **df.AtomTypes.buried**: number of buried AtomTypes for each structure
- **df.AtomTypes.SurExp**: number of surface exposed AtomTypes for each structure
- **df.AtomTypes.h2o.nearby**: number of surface exposed AtomTypes within h2o.prot.dist.max (default 6 Ang) of an individual water
- **df.AtomTypes.h2o.bound**: number of surface exposed AtomTypes within bound.h2o.dist.max (default 4 Ang) of an indvidual water
- **df.AtomTypes.h2o.inter**: number of surface exposed AtomTypes with the shortest distance to an individual water
- **df.residue.hydro**:
- **HydrophilicityTable**: hydrophilicity table based on provided protein structures
- **AtomTypeClasses.hydratFract**:

- **no.h2o**: proteins (PDB IDs) without the *minimum* number of user defined waters `min.num.h2o`
- **call**: parameters provided by the user
- **duration**: duration of complete `HydrophilicityEvaluation()` calculation

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Leslie A Kuhn, Craig A Swanson, Michael E Pique, John A Tainer, and Elizabeth D Getzof. Atomic and Residue Hydrophilicity in the Context of Folded Protein Structures. *PROTEINS: Structure, Function, and Genetics*, 1995, **23** (*4*), pp 536-547. DOI: 10.1002/prot.340230408 PMID: 8749849

See Also

Other "Hydrophilicity Evaluation" "Bound Water Environment": `calcAtomClassHydrophilicity`, `calcAtomHydrationEstimate`, `getProtAtomsNearWater`, `getResidueData`

Examples

```
## Not run:
 HydrophilicityEvaluation <- function(prefix = "alignTesting/",
                                      h2o.prot.dist.max = 6.0,
                                      bound.h2o.dist.max = 4.0,
                                      min.num.h2o = 20,
                                      probeRadius = 1.4,
                                      dataset = "top56")

## End(Not run)
```

HydrophilicityTable *Residue Atom Type Hydrophilicity Values*

Description

Atomic hydrophilicity values for the 20 naturally occurring amino acids and water.

Details

The Hydrophilicity Table is based on the work of Esposito (see reference below) in vanddraabe package. The hydrophilicity values are based on information from a 1995 analysis of published PDB structures and indicate how likely the individual atoms of the amino acid residues are to have a water molecule within 4.0 angstroms.

The data contained within the Hydrophilicity Table is based on ~7900 experimentally determined crystallographic protein structures with resolution values less than or equal to x.x Angstroms, a R-factor less than or equal to 0.26, and 20 or more bound waters each. The protein structures are from

the Top8000 ("a database of about 8000 high-resolution, quality-filtered protein chains"; reference below) high quality protein dataset from the Kinemage laboratory at Duke University. The included structures had a range of B-values and occupancy values.

These values are based on the methods and protocols of Kuhn *et al*.

The Hydrophilicity Table contains:

- **residueAtomName**: Contracted residue type and atom name to aid looking up hydrophilicity values.
- **residue**: Three-letter residue name.
- **atomName**: Atom name indicating the atom type and its position in the amino acid residue.
- **surfaceOccurrences**: Number of times each atom has a defined solvent exposed surface area.
- **hydratOccurrences**: Proportion of the solvent exposed residue-specific atom type with a water molecule closely bound (within 4.0 Angstroms).

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Leslie A Kuhn, Craig A Swanson, Michael E Pique, John A Tainer, and Elizabeth D Getzof. Atomic and Residue Hydrophilicity in the Context of Folded Protein Structures. *PROTEINS: Structure, Function, and Genetics*, 1995, **23** (*4*), pp 536-547. DOI: 10.1002/prot.340230408 PMID: 8749849

Bradley J Hintze, Steven M Lewis, Jane S Richardson, and David C Richardson. Molprobity's ultimate rotamer-library distributions for model validation. *Proteins: Structure, Function, and Bioinformatics*, 2016, **84** (*9*), pp 1177-1189. DOI: 10.1002/prot.25039 Top8000 webpage

---

`Mobility` *Water Molecule Mobility*

Description

Calculate the mobility values of waters for a structure.

Usage

```
Mobility(Bvalues, occupancy)
```

Arguments

`Bvalues`	B-value values from the imported PDB file(s)
`occupancy`	Occumpancy values from the imported PDB file(s)

Details

The mobility of waters within a structure is normalization method to identify the amount of variance an atom has within a structure. In the case of waters, identified by an oxygen atom without hydrogen atoms, a water-oxygen atom with a mobility value of 0 is considered rigid and does not possess variance. The average mobility within a structure has value of 1 while an atom's mobility value of x is considered x-times as mobile as an average atom.

$$Mobility = \frac{\frac{B-value}{\mu_{B-value}}}{\frac{Occupancy}{\mu_{Occupancy}}}$$

Mobility is calculated using the B-value and occupancy values; these values are a byproduct of solving the 3D molecular structure from electron density maps. The mobility values allows us to compare atomic mobility between molecular structures solved using different structural refinement methods. Atoms, in this instance water-oxygens, with a mobility value greater than 2.0 are removed from analysis.

The mobility exclusion value is user defined within the main `ConservedWaters()` function but has a default value of 2.0.

Value

Vector of mobility values; unitless.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

See Also

Other "Bound Water Environment": `BoundWaterEnvironment.interact`, `BoundWaterEnvironment.quality`, `BoundWaterEnvironment`, `NormalizedBvalue`, `calcBvalue`, `calcNearbyHydrationFraction`, `calcNumHydrogenBonds`

Examples

```
set.seed(13)
sample.idc <- sample(1:nrow(thrombin.1hai$atom), 10)
Bvalues <- thrombin.1hai$atom[sample.idc, "b"]
Bvalues
# [1] 45.73 45.40 20.24 39.30 35.53
#     22.16 35.81 15.35 22.73 21.34
occupancy <- thrombin.1hai$atom[sample.idc, "o"]
occupancy
# [1] 0.01 1.00 1.00 1.00 1.00
```

```
#     1.00 1.00 1.00 1.00 1.00
Mobility(Bvalues, occupancy)
# [1] 135.7183 1.3474 0.6007 1.1664 1.0545
#       0.6577 1.0628 0.4556 0.6746 0.6333
```

`MobilityBarplot` *Mobility Barplots*

Description

Mobility Barplots for Cluster with at least 50% Conservation

Usage

```
MobilityBarplot(data, passed.waters = TRUE)
```

Arguments

`data`	The `h2o.clusters.summary` data.frame from the `ClusterWaters()` function containing the `mobility.mu` information.
`passed.waters`	Logical indicator to plot results for waters **passing** `Mobility()` and `NormalizedBvalue()` **OR** using **all** waters within the PDB files.

Details

Constructs a barplot with corresponding density plot for the mean mobility value for all water within each cluster with at least 50% water conservation. Clusters with 50 to 69% water conseration are dark red, clusters with 70 to 79% conservation are red, 80 to 89% conservation are light blue, 90 to 99% conservation are blue, and 100% conservation (waters from all structures) are dark blue.

The mobility values are calculated by the `Mobility()` function.

This plot was inspired by Figure 1 of Sanschagrin and Kuhn (1998).

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

See Also

Other plots: `BoundWaterEnvPlots`, `BoundWaterEnvSummaryPlot`, `BvalueBarplot.summ`, `BvalueBarplot`, `ClusterSummaryPlots`, `MobNormBvalEvalPlots`, `MobilityBarplot.summ`, `OccupancyBarplot.summ`, `OccupancyBarplot`, `nBvalueBarplot`, `normBvalueBarplot.summ`

Examples

```
## Not run:
mobility.plot <- MobilityBarplot(data=thrombin10.conservedWaters,
                                 passed.waters=TRUE)

## End(Not run)
```

`MobilityBarplot.summ` *Mobility Summary Barplots*

Description

Mobility summary barplots for PDB structures. The plots are faceted and displays the binned B-value values for all the structures. The counts are presented on a `log10` scale. The function will automatically plot ten plots per page.

Usage

```
MobilityBarplot.summ(data)
```

Arguments

`data`	The results from the `CleanProteinStructures()` function. Will use the binned mobility data.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other plots: `BoundWaterEnvPlots`, `BoundWaterEnvSummaryPlot`, `BvalueBarplot.summ`, `BvalueBarplot`, `ClusterSummaryPlots`, `MobNormBvalEvalPlots`, `MobilityBarplot`, `OccupancyBarplot.summ`, `OccupancyBarplot`, `nBvalueBarplot`, `normBvalueBarplot.summ`

Examples

```
## Not run:
MobilityBarplot.summ(data)

##----- multiple pages
library(ggforce)
mob.barplots.summary <- MobilityBarplot.summ(data)
num.pages <- ceiling(nrow(data$mobility.counts) / 10)

pdf(file="multiple_pages.pdf", height=11, width=8.5)
for (page in seq_len(num.pages)) {
  print(mob.barplots.summary +
```

```
        ggforce::facet_wrap_paginate(~PDBid,
                                     ncol = 2, nrow = 5, page = page) )
  }
  dev.off()

## End(Not run)
```

`MobNormBvalEvalPlots` *Mobility and Normalized B-values Evaluation Plots*

Description

Mean bound water environment summary per percent conservation

Usage

```
MobNormBvalEvalPlots(data, passed.waters = TRUE,
  title = "Mobility and Normalized B-value Evaluation")
```

Arguments

`data`	The `h2o.clusters.summary` data.frame from the `ConservedWaters()` function containing the `nBvalue.mu` information. This data.frame is found within the `h2o.cluster.passed` and `h2o.cluster.all`
`passed.waters`	Logical indicator to plot results for waters **passing** `Mobility()` and `NormalizedBvalue()` **OR** using **all** waters within the `PDB` files.
`title`	The title for the plot

Details

Constructs a series of scatterplots illustrating the relationship between mobility and normalized B-values and (i) percent water conservation, (ii) mean distance between waters in a cluster, and (iii) mean distance between waters in a cluster and the cluster's centroid. The dots are colored based on water cluster "percent conservation":

- **light grey dots**: less than 50% conservation
- **dark red dots**: 50% to 69% conservations
- **red dots**: 70% to 79% conservation
- **light blue dots**: 80% to 89% conservation
- **blue dots**: 90% to 99% conservation
- **dark blue dots**: 100% conservation (all structures contribute to the water cluster).

The mean distance plots will have a column of dots at a distance of 0.0 if there are clusters composed of a single water molecule. Thus, these clusters have a zero distance between and to other waters in their cluster because there are **no other waters** in their cluster.

This plot was inspired by Figure 2 of Ogata and Wodak (2002).

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Koji Ogata and Shoshana J Wodak. Conserved water molecules in MHC class-I molecules and their putative structural and functional roles. *Protein Engineering*, 2002, **15** (*8*), pp 697-705. DOI: 10.1093/protein/15.8.697 PMID: 12364585

See Also

Other plots: BoundWaterEnvPlots, BoundWaterEnvSummaryPlot, BvalueBarplot.summ, BvalueBarplot, ClusterSummaryPlots, MobilityBarplot.summ, MobilityBarplot, OccupancyBarplot.summ, OccupancyBarplot, nBvalueBarplot, normBvalueBarplot.summ

Examples

```
## Not run:
bwe.summary.plot <- MobNormBvalEvalPlots(data=thrombin10.conservedWaters,
                                         passed.waters=TRUE,
                                    title="Mobility and Normalized B-value Evaluation")

## End(Not run)
```

---

names.backbone.atoms *Backbone Atom Names*

Description

Backbone atom names based on PDB atom naming conventions.

Usage

```
names.backbone.atoms
```

Format

An object of class character of length 4.

Details

Protein backbone atom names based on the PDB atom naming conventions.

- **N**: Nitrogen backbone atom; amide, "leading" functional group
- **CA**: alpha-Carbon backbone atom; bonds/connects the side chain to the backbone
- **C**: Carbon backbone atom; carboxyl, "tail" functional group
- **O**: Oxygen backbone atom double bonded to the carbon backbone (C) atom; part of the carboxyl, "tail" functional groups

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: names.polar.atoms, names.res.AtomTypes, names.resATs.carb.sulf, names.resATs.nitro.neut, names.resATs.nitro.pos, names.resATs.oxy.neg, names.resATs.oxy.neut, names.residues, names.sidechain.atoms, names.waters

Examples

```
names.backbone.atoms
# [1] "N"  "CA" "C"  "O"
```

---

names.polar.atoms *Polar Atom Names*

Description

Polar atom names based on PDB atom naming conventions.

Usage

```
names.polar.atoms
```

Format

An object of class character of length 20.

Details

Polar atoms are those possessing a lone pair(s) of elections able to participate in hydrogen bonds with hydrogen atoms within 3.5 Angstroms and XX degrees of the lone pair containing atom. Traditionally, nitrogen, oxygen, and sulfur atoms possess lone pair(s) of electrons and participate in hydrogen bonds in biological sytems. Water molecules are able to hydrogen bond with and participate in hydrogen bonds.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: names.backbone.atoms, names.res.AtomTypes, names.resATs.carb.sulf, names.resATs.nitro.neut, names.resATs.nitro.pos, names.resATs.oxy.neg, names.resATs.oxy.neut, names.residues, names.sidechain.atoms, names.waters

Examples

```
names.polar.atoms
# [1] "N"   "NE"  "NH1" "NH2" "ND2" "NE2" "ND1" "NZ"  "NE1" "O"   "OD1"
#     "OD2" "OE1" "OE2" "OG"  "OG1" "OH"  "S"   "SD"  "SG"
```

---

names.res.AtomTypes *Residue and AtomType Names*

Description

Residue and AtomType names based on PDB atom naming conventions.

Usage

```
names.res.AtomTypes
```

Format

An object of class `character` of length 167.

Details

The 167 residue-atomtype names based on the 20 naturally occurring amino acids.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: `names.backbone.atoms`, `names.polar.atoms`, `names.resATs.carb.sulf`, `names.resATs.nitro.neut`, `names.resATs.nitro.pos`, `names.resATs.oxy.neg`, `names.resATs.oxy.neut`, `names.residues`, `names.sidechain.atoms`, `names.waters`

Examples

```
names.res.AtomTypes
# [1] "ALA C"   "ALA CA"  "ALA CB"  "ALA N"   "ALA O"   "ARG C"   "ARG CA"
#     "ARG CB"  "ARG CD"  "ARG CG"  "ARG CZ"  "ARG N"   "ARG NE"  "ARG NH1"
#     "ARG NH2" "ARG O"   "ASN C"   "ASN CA"  "ASN CB"  "ASN CG"  "ASN N"
#     "ASN ND2" "ASN O"   "ASN OD1" "ASP C"   "ASP CA"  "ASP CB"  "ASP CG"
#     "ASP N"   "ASP O"   "ASP OD1" "ASP OD2" "CYS C"   "CYS CA"  "CYS CB"
#     "CYS N"   "CYS O"   "CYS SG"  "GLN C"   "GLN CA"  "GLN CB"  "GLN CD"
#     "GLN CG"  "GLN N"   "GLN NE2" "GLN O"   "GLN OE1" "GLU C"   "GLU CA"
#     "GLU CB"  "GLU CD"  "GLU CG"  "GLU N"   "GLU O"   "GLU OE1" "GLU OE2"
#     "GLY C"   "GLY CA"  "GLY N"   "GLY O"   "HIS C"   "HIS CA"  "HIS CB"
#     "HIS CD2" "HIS CE1" "HIS CG"  "HIS N"   "HIS ND1" "HIS NE2" "HIS O"
#     "ILE C"   "ILE CA"  "ILE CB"  "ILE CD1" "ILE CG1" "ILE CG2" "ILE N"
```

```
#     "ILE O"   "LEU C"   "LEU CA"  "LEU CB"  "LEU CD1" "LEU CD2" "LEU CG"
#     "LEU N"   "LEU O"   "LYS C"   "LYS CA"  "LYS CB"  "LYS CD"  "LYS CE"
#     "LYS CG"  "LYS N"   "LYS NZ"  "LYS O"   "MET C"   "MET CA"  "MET CB"
#     "MET CE"  "MET CG"  "MET N"   "MET O"   "MET SD"  "PHE C"   "PHE CA"
#     "PHE CB"  "PHE CD1" "PHE CD2" "PHE CE1" "PHE CE2" "PHE CG"  "PHE CZ"
#     "PHE N"   "PHE O"   "PRO C"   "PRO CA"  "PRO CB"  "PRO CD"  "PRO CG"
#     "PRO N"   "PRO O"   "SER C"   "SER CA"  "SER CB"  "SER N"   "SER O"
#     "SER OG"  "THR C"   "THR CA"  "THR CB"  "THR CG2" "THR N"   "THR O"
#     "THR OG1" "TRP C"   "TRP CA"  "TRP CB"  "TRP CD1" "TRP CD2" "TRP CE2"
#     "TRP CE3" "TRP CG"  "TRP CH2" "TRP CZ2" "TRP CZ3" "TRP N"   "TRP NE1"
#     "TRP O"   "TYR C"   "TYR CA"  "TYR CB"  "TYR CD1" "TYR CD2" "TYR CE1"
#     "TYR CE2" "TYR CG"  "TYR CZ"  "TYR N"   "TYR O"   "TYR OH"  "VAL C"
#     "VAL CA"  "VAL CB"  "VAL CG1" "VAL CG2" "VAL N"   "VAL O"
```

`names.resATs.carb.sulf`

Carbon and Sulfur Residue-AtomType Names

Description

Carbon and sulfur residue-atomtype names based on PDB atom naming conventions.

Usage

```
names.resATs.carb.sulf
```

Format

An object of class `character` of length 109.

Details

These residue-atomtype names indicate carbon and sulfur atoms with a neutral charge.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: `names.backbone.atoms`, `names.polar.atoms`, `names.res.AtomTypes`, `names.resATs.nitro.neut`, `names.resATs.nitro.pos`, `names.resATs.oxy.neg`, `names.resATs.oxy.neut`, `names.residues`, `names.sidechain.atoms`, `names.waters`

Examples

```
names.resATs.carb.sulf
# [1] "ALA CA"  "ALA C"   "ALA CB"  "ARG CA"  "ARG C"   "ARG CB"  "ARG CG"
# "ARG CD"  "ARG CZ"  "ASN CA"  "ASN C"   "ASN CB"  "ASN CG"  "ASP CA"  "ASP C"
# [16] "ASP CB"  "ASP CG"  "CYS CA"  "CYS C"   "CYS CB"  "CYS SG"  "GLN CA"
# "GLN C"   "GLN CB"  "GLN CG"  "GLN CD"  "GLU CA"  "GLU C"   "GLU CB"  "GLU CG"
# [31] "GLU CD"  "GLY CA"  "GLY C"   "HIS CA"  "HIS C"   "HIS CB"  "HIS CG"
# "HIS CD2" "HIS CE1" "ILE CA"  "ILE C"   "ILE CB"  "ILE CG1" "ILE CG2" "ILE CD1"
# [46] "LEU CA"  "LEU C"   "LEU CB"  "LEU CG"  "LEU CD1" "LEU CD2" "LYS CA"
# "LYS C"   "LYS CB"  "LYS CG"  "LYS CD"  "LYS CE"  "MET CA"  "MET C"   "MET CB"
# [61] "MET CG"  "MET SD"  "MET CE"  "PHE CA"  "PHE C"   "PHE CB"  "PHE CG"
# "PHE CD1" "PHE CD2" "PHE CE1" "PHE CE2" "PHE CZ"  "PRO CA"  "PRO C"   "PRO CB"
# [76] "PRO CG"  "PRO CD"  "SER CA"  "SER C"   "SER CB"  "THR CA"  "THR C"
# "THR CB"  "THR CG2" "TRP CA"  "TRP C"   "TRP CB"  "TRP CG"  "TRP CD1" "TRP CD2"
# [91] "TRP CE2" "TRP CE3" "TRP CZ2" "TRP CZ3" "TRP CH2" "TYR CA"  "TYR C"
# "TYR CB"  "TYR CG"  "TYR CD1" "TYR CD2" "TYR CE1" "TYR CE2" "TYR CZ"  "VAL CA"
# [106] "VAL C"   "VAL CB"  "VAL CG1" "VAL CG2"
```

---

names.resATs.nitro.neut

Neutral Nitrogen Residue-AtomType Names

Description

Neutral nitrogen residue-atomtype names based on PDB atom naming conventions.

Usage

```
names.resATs.nitro.neut
```

Format

An object of class `character` of length 24.

Details

These residue-atomtype names indicate nitrogen atoms with a neutral charge.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: `names.backbone.atoms`, `names.polar.atoms`, `names.res.AtomTypes`, `names.resATs.carb.sulf`, `names.resATs.nitro.pos`, `names.resATs.oxy.neg`, `names.resATs.oxy.neut`, `names.residues`, `names.sidechain.atoms`, `names.waters`

Examples

```
names.resATs.nitro.neut
# [1] "ALA N"   "ARG N"   "ARG NE"  "ASN N"   "ASN ND2" "ASP N"   "CYS N"
# "GLN N"   "GLN NE2" "GLU N"   "GLY N"   "HIS N"   "ILE N"   "LEU N"   "LYS N"
# [16] "MET N"   "PHE N"   "PRO N"   "SER N"   "THR N"   "TRP N"   "TRP NE1"
# "TYR N"   "VAL N"
```

---

names.resATs.nitro.pos

Positive Nitrogen Residue-AtomType Names

Description

Positive nitrogen residue-atomtype names based on PDB atom naming conventions.

Usage

```
names.resATs.nitro.pos
```

Format

An object of class `character` of length 5.

Details

These residue-atomtype names indicate nitrogen atoms with a positive charge.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: `names.backbone.atoms`, `names.polar.atoms`, `names.res.AtomTypes`, `names.resATs.carb.sulf`, `names.resATs.nitro.neut`, `names.resATs.oxy.neg`, `names.resATs.oxy.neut`, `names.residues`, `names.sidechain.atoms`, `names.waters`

Examples

```
names.resATs.nitro.pos
# [1] "ARG NH1" "ARG NH2" "HIS ND1" "HIS NE2" "LYS NZ"
```

names.resATs.oxy.neg *Negative Oxygen Residue-AtomType Names*

Description

Negatvie oxygen residue-atomtype names based on PDB atom naming conventions.

Usage

```
names.resATs.oxy.neg
```

Format

An object of class `character` of length 4.

Details

These residue-atomtype names indicate oxygen atoms with a negative charge.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: `names.backbone.atoms`, `names.polar.atoms`, `names.res.AtomTypes`, `names.resATs.carb.sulf`, `names.resATs.nitro.neut`, `names.resATs.nitro.pos`, `names.resATs.oxy.neut`, `names.residues`, `names.sidechain.atoms`, `names.waters`

Examples

```
names.resATs.oxy.neg
# [1] "ASP OD1" "ASP OD2" "GLU OE1" "GLU OE2"
```

names.resATs.oxy.neut *Neutral Oxygen Residue-AtomType Names*

Description

Neutral oxygen residue-atomtype names based on PDB atom naming conventions.

Usage

```
names.resATs.oxy.neut
```

Format

An object of class `character` of length 25.

Details

These residue-atomtype names indicate oxygen atoms with a neutral charge.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: `names.backbone.atoms`, `names.polar.atoms`, `names.res.AtomTypes`, `names.resATs.carb.sulf`, `names.resATs.nitro.neut`, `names.resATs.nitro.pos`, `names.resATs.oxy.neg`, `names.residues`, `names.sidechain.atoms`, `names.waters`

Examples

```
names.resATs.oxy.neut
# [1] "ALA O"   "ARG O"   "ASN O"   "ASN OD1" "ASP O"   "CYS O"   "GLN O"
# "GLN OE1" "GLU O"   "GLY O"   "HIS O"   "ILE O"   "LEU O"   "LYS O"   "MET O"
# [16] "PHE O"   "PRO O"   "SER O"   "SER OG"  "THR O"   "THR OG1" "TRP O"
# "TYR O"   "TYR OH"  "VAL O"
```

---

names.residues *Residue Names*

Description

Residue names based on PDB atom naming conventions.

Usage

```
names.residues
```

Format

An object of class `character` of length 20.

Details

The three (3) letter abbreviation for the twenty (20) naturally occurring amino acid residues.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: names.backbone.atoms, names.polar.atoms, names.res.AtomTypes, names.resATs.carb.sulf, names.resATs.nitro.neut, names.resATs.nitro.pos, names.resATs.oxy.neg, names.resATs.oxy.neut, names.sidechain.atoms, names.waters

Examples

```
names.residues
# [1] "ALA" "ARG" "ASN" "ASP" "CYS" "GLN" "GLU" "GLY" "HIS" "ILE"
#     "LEU" "LYS" "MET" "PHE" "PRO" "SER" "THR" "TRP" "TYR" "VAL"
```

---

names.sidechain.atoms *Sidechain Atom Names*

Description

Sidechain atom names based on PDB atom naming conventions.

Usage

```
names.sidechain.atoms
```

Format

An object of class character of length 32.

Details

The 32 unique sidechain atom names. The first character is the element and the second character is the Greek letter (B=beta, D=delta, E=epsilon, G=gamma, Z=zeta) defining the specific position within the sidechain. The exception to the use of Greek letters is OH indicating a hydroxyl group at the *para* position of the six-member ring of tyrosine. Some sidechain atom names have a number in the third character position when there are mirrored/symetrical atoms; *e.g.*, CG1 and CG2 of valine.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: names.backbone.atoms, names.polar.atoms, names.res.AtomTypes, names.resATs.carb.sulf, names.resATs.nitro.neut, names.resATs.nitro.pos, names.resATs.oxy.neg, names.resATs.oxy.neut, names.residues, names.waters

Examples

```
names.sidechain.atoms
# [1] "CB"  "CG"  "CD"  "NE"  "CZ"  "NH1" "NH2" "OD1" "ND2" "OD2" "SG"
#     "OE1" "NE2" "OE2" "CD2" "ND1" "CE1" "CG1" "CG2" "CD1" "CE"  "NZ"
#     "SD"  "CE2" "OG"  "OG1" "NE1" "CE3" "CZ2" "CZ3" "CH2" "OH"
```

---

names.waters *Water Residue Names*

Description

Water residue names based on PDB naming conventions.

Usage

```
names.waters
```

Format

An object of class `character` of length 3.

Details

The three (3) letter abbreviation for the three (3) commonly used abbreviations for water residues.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other constants: `names.backbone.atoms`, `names.polar.atoms`, `names.res.AtomTypes`, `names.resATs.carb.sulf`, `names.resATs.nitro.neut`, `names.resATs.nitro.pos`, `names.resATs.oxy.neg`, `names.resATs.oxy.neut`, `names.residues`, `names.sidechain.atoms`

Examples

```
names.waters
# [1] "HOH", "DOD", "WAT"
```

nBvalueBarplot *Normalized B-value Barplots*

Description

Normalized B-value Barplots for Cluster with at least 50% Conservation

Usage

```
nBvalueBarplot(data, passed.waters = TRUE)
```

Arguments

data	The h2o.clusters.summary data.frame from the ClusterWaters() function containing the nBvalue.mu information.
passed.waters	Logical indicator to plot results for waters **passing** Mobility() and NormalizedBvalue() **OR** using **all** waters within the PDB files.

Details

Constructs a barplot with corresponding density plot for the mean normalized B-value value for all water within each cluster with at least 50% water conservation. Clusters with 50 to 69% water conseration are dark red, clusters with 70 to 79% conservation are red, 80 to 89% conservation are light blue, 90 to 99% conservation are blue, and 100% conservation (waters from all structures) are dark blue.

The normalized B-value values are calculated by the NormalizedBvalue() function.

This plot was inspired by Figure 1 of Sanschagrin and Kuhn (1998).

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

See Also

Other plots: BoundWaterEnvPlots, BoundWaterEnvSummaryPlot, BvalueBarplot.summ, BvalueBarplot, ClusterSummaryPlots, MobNormBvalEvalPlots, MobilityBarplot.summ, MobilityBarplot, OccupancyBarplot.summ, OccupancyBarplot, normBvalueBarplot.summ

Examples

```
## Not run:
nBvalue.plot <- nBvalueBarplot(data=thrombin10.conservedWaters,
                               passed.waters=TRUE)

## End(Not run)
```

Nearby *Nearby*

Description

Determine the entities near the entity of interest using a distance matrix.

Usage

```
Nearby(distances, set.idc, radius = 3.6)
```

Arguments

distances	Vector of distance values; see above note.
set.idc	Vector of indices (as integers) indicating the entities of interest. This set of entities corresponds to the columns of the distance matrix because the provided distance matrix should be square. No check is performed on the squareness of the distance matrix because it is calculated **within** the ConservedWaters function.
radius	Numerical value indicating the distance to look for neighboring entities; default: 3.6

Details

Identify the entity, or entities, near an entity or collection of entites of interest. The previously calculated distance matrix, set of indicies, and a user defined radius are required.

NOTE: This function is designed to work with `BoundWaterEnvironment()` and the `base::apply()` function processing rows (the `MARGIN = 1` option). For this reason it is **NOT** a public function.

Value

Vector of indicies.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
## Not run:
##----- determine atom indices
ProtHetWat.idc <- ProtHetWatIndices(thrombin.1hai$atom)
prot.idc <- ProtHetWat.idc$prot.idc
het.idc <- ProtHetWat.idc$het.idc
h2o.idc <- ProtHetWat.idc$h2o.idc

##----- calculate the distances
atoms.dist <- as.matrix(dist(thrombin.1hai$atom[, c("x","y","z")],
                             method = "euclidean",
                             diag = TRUE, upper = TRUE))
diag(atoms.dist) <- NA
atom.idc <- sort(c(prot.idc, het.idc, h2o.idc))
atoms.dist <- atoms.dist[atom.idc, atom.idc]

##----- determine nearby atoms
nearby.prot.idc <- Nearby(distances = atoms.dist[h2o.idc[1], ],
                          set.idc = prot.idc,
                          radius = 3.6)
nearby.prot.idc
# [1] 571
atoms.dist[h2o.idc[1], nearby.prot]
# [1] 3.571

## End(Not run)
```

NormalizedBvalue *B-value Normalization*

Description

Calculate the normalized B-value values of waters for a structure.

Usage

```
NormalizedBvalue(Bvalues)
```

Arguments

Bvalues	B-value values

Details

The normalized B-value values are the number of standard deviations from the mean for the water oxygens' B-values within the structure of interest.

The B-value normalization exclusion value is user defined within the main ConservedWaters() function but has a default value of 1.0.

Value

Vector of normalized and unitless B-value values.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Oliviero Carugo. Correlation between occupancy and B value of water molecules in protein crystal structures. *Protein Engineering*, 1999, **12** (*12*), pp 1021-1024. DOI: 10.1093/protein/12.12.1021 PMID: 10611392

See Also

Other "Bound Water Environment": BoundWaterEnvironment.interact, BoundWaterEnvironment.quality, BoundWaterEnvironment, Mobility, calcBvalue, calcNearbyHydrationFraction, calcNumHydrogenBonds

Examples

```
set.seed(13)
Bvalues <- sample(thrombin.1hai$atom$b, 10)
Bvalues
# [1] 45.73 45.40 20.24 39.30 35.53
#     22.16 35.81 15.35 22.73 21.34
NormalizedBvalue(Bvalues)
# [1]  1.3698  1.3404 -0.9017  0.7968  0.4608
#     -0.7306  0.4858 -1.3375 -0.6798 -0.8037
```

normBvalueBarplot.summ

Normalized B-value Summary Barplots

Description

B-value summary barplots for the PDB structures. The plots are faceted and displays the binned B-value values for all the structures. The counts are presented on a log10 scale.

Usage

```
normBvalueBarplot.summ(data)
```

Arguments

data	The results from the CleanProteinStructures() function. Will use the binned normalized B-value data.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other plots: BoundWaterEnvPlots, BoundWaterEnvSummaryPlot, BvalueBarplot.summ, BvalueBarplot, ClusterSummaryPlots, MobNormBvalEvalPlots, MobilityBarplot.summ, MobilityBarplot, OccupancyBarplot.summ, OccupancyBarplot, nBvalueBarplot

Examples

```
## Not run:
normBvalueBarplot.summ(data)

##----- multiple pages
library(ggforce)
nBvalue.barplots.summary <- normBvalueBarplot.summ(data)
num.pages <- ceiling(nrow(data$normBvalue.counts) / 10)

pdf(file="multiple_pages.pdf", height=11, width=8.5)
for (page in seq_len(num.pages)) {
  print(nBvalue.barplots.summary +
        ggforce::facet_wrap_paginate(~PDBid,
                                     ncol = 2, nrow = 5, page = page) )
}
dev.off()

## End(Not run)
```

OccupancyBarplot *Occupancy Barplots*

Description

Occupancy Barplots for Cluster with at least 50% Conservation

Usage

```
OccupancyBarplot(data, passed.waters = TRUE)
```

Arguments

data	The h2o.clusters.summary data.frame from the ClusterWaters() function containing the o.mu information.
passed.waters	Logical indicator to plot results for waters **passing** Mobility() and NormalizedBvalue() **OR** using **all** waters within the PDB files.

Details

Constructs a barplot with corresponding density plot for the mean occupancy value for all water within each cluster with at least 50% water conservation. Clusters with 50 to 69% water conseration are dark red, clusters with 70 to 79% conservation are red, 80 to 89% conservation are light blue, 90 to 99% conservation are blue, and 100% conservation (waters from all structures) are dark blue.

This plot was inspired by Figure 1 of Sanschagrin and Kuhn (1998).

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

Paul C Sanschagrin and Leslie A Kuhn. Cluster analysis of consensus water sites in thrombin and trypsin shows conservation between serine proteases and contributions to ligand specificity. *Protein Science*, 1998, **7** (*10*), pp 2054-2064. DOI: 10.1002/pro.5560071002 PMID: 9792092 WatCH webpage

See Also

Other plots: BoundWaterEnvPlots, BoundWaterEnvSummaryPlot, BvalueBarplot.summ, BvalueBarplot, ClusterSummaryPlots, MobNormBvalEvalPlots, MobilityBarplot.summ, MobilityBarplot, OccupancyBarplot.summ, nBvalueBarplot, normBvalueBarplot.summ

Examples

```
## Not run:
occupancy.plot <- OccupancyBarplot(data=thrombin10.conservedWaters,
                                   passed.waters=TRUE)

## End(Not run)
```

`OccupancyBarplot.summ` *Occupancy Summary Barplots*

Description

Occupancy summary barplots for the PDB structures. The plots are faceted and displays the binned occupancy values for all the structures. The counts are presented on a `log10` scale.

Usage

```
OccupancyBarplot.summ(data)
```

Arguments

`data`	The results from the `CleanProteinStructures()` function. Will use the binned occupancy data.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other plots: `BoundWaterEnvPlots`, `BoundWaterEnvSummaryPlot`, `BvalueBarplot.summ`, `BvalueBarplot`, `ClusterSummaryPlots`, `MobNormBvalEvalPlots`, `MobilityBarplot.summ`, `MobilityBarplot`, `OccupancyBarplot`, `nBvalueBarplot`, `normBvalueBarplot.summ`

Examples

```
## Not run:
OccupancyBarplot.summ(data)

##----- multiple pages
library(ggforce)
occ.barplots.summary <- OccupancyBarplot.summ(data)
num.pages <- ceiling(nrow(data$occupancy.counts) / 10)

pdf(file="multiple_pages.pdf", height=11, width=8.5)
for (page in seq_len(num.pages)) {
  print(occ.barplots.summary +
        ggforce::facet_wrap_paginate(~PDBid,
```

```
                                        ncol = 2, nrow = 5, page = page) )
  }
  dev.off()

## End(Not run)
```

`openxlsxCellStyles` *openxlsx Cell Style*

Description

A collection of cell style formats for the openxlsx package.

Details

A centralized location defining the cell styles removes the need to change the formatting in several functions and provides a way to standardize cell formatting throughout the results.

The cell styles for the openxlsx package are defined within the `openxlsxCellStyle.R` file.

The defined cell styles are:

- **cs.green**: background: lime, font: green and bold
- **cs.pink**: background: pink, font: red and bold
- **cs.amber**: background: amber, font orange and bold
- **cs.0digits**: integer?
- **cs.comma**: comma delineated values; *e.g.*, `1,234`
- **cs.date**: date formatted
- **cs.1digits**: one digit after the decimal point
- **cs.2digits**: two digits after the decimal point
- **cs.3digits**: three digits after the decimal point
- **cs.4digits**: four digits after the decimal point
- **cs.header**: top row of table; font: black, bold, centered, with a line along the bottom of the cell
- **cs.titles.tables**: top row of table; font: black, bold, and centered

Author(s)

Emilio Xavier Esposito `<emilio@exeResearch.com>`

See Also

Other "openxlsx functions": `oxAlignOverlapSheet`, `oxClusterStatsSheet`, `oxClusterSummarySheet`, `oxInitWaterDataSheet`, `oxPDBcleanedSummarySheet`, `oxPlainDataSheet`, `oxRCSBinfoSheet`, `oxWaterOccurrenceSheet`

`oxAlignOverlapSheet` *Align Overlap Data Sheet*

Description

Constructs the openxlsx worksheet for the `AlignOverlap()` results.

Usage

```
oxAlignOverlapSheet(wb.name, sheet.name = "AlignOverlap", df)
```

Arguments

`wb.name`	Name of the workbook for the results; *e.g.*, results.wb
`sheet.name`	Name of the worksheet being formatted; default: `"AlignOverlap"`
`df`	data.frame containing the summary of `AlignOverlap()`; *e.g.*, `df.results`

Details

This function is to *ONLY* be used with the results of `AlignOverlap()`. Specific aspects of how the returned `data.frame` will be formatted are **hard-coded** into this function.

Notable formatting:

- Top row frozen
- Column widths are set based on column content
- Structures passing the `AlignOverlap()` evaluation are hightlighted lime green
- Structures failing the `AlignOverlap()` evaluation are hightlighted pink

This openxlsx function is **NOT** exported.

Value

The workbook containing the indicated and newly formatted worksheet.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "openxlsx functions": `openxlsxCellStyles`, `oxClusterStatsSheet`, `oxClusterSummarySheet`, `oxInitWaterDataSheet`, `oxPDBcleanedSummarySheet`, `oxPlainDataSheet`, `oxRCSBinfoSheet`, `oxWaterOccurrenceSheet`

`oxClusterStatsSheet` *openxlsx Water Cluster Statistics*

Description

Constructs the openxlsx worksheet for the Water Cluster statistics.

Usage

```
oxClusterStatsSheet(wb.name, sheet.name = "ClusterStatistics", df)
```

Arguments

`wb.name`	Name of the workbook for the results; *e.g.*, results.wb
`sheet.name`	Name of the worksheet being formatted; default: "ClusterStatistics"
`df`	data.frame containing the results of the `GetSimilarityPairs` function; *e.g.*, `h2o.cluster.stats`

Details

This function is to *ONLY* be used with the results of `ConservedWaterStats()`. Specific aspects of how the returned `data.frame` will be formatted are **hard-coded** into this function.

This openxlsx function is **NOT** exported.

Value

The workbook containing the indicated and newly formatted worksheet.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "openxlsx functions": `openxlsxCellStyles`, `oxAlignOverlapSheet`, `oxClusterSummarySheet`, `oxInitWaterDataSheet`, `oxPDBcleanedSummarySheet`, `oxPlainDataSheet`, `oxRCSBinfoSheet`, `oxWaterOccurrenceSheet`

`oxClusterSummarySheet` *openxlsx Cluster Summary Sheet*

Description

Constructs the openxlsx worksheet for the Cluster Summary analysis.

Usage

```
oxClusterSummarySheet(wb.name, sheet.name = "ClusterSummary", df)
```

Arguments

`wb.name`	Name of the workbook for the results; *e.g.*, results.wb
`sheet.name`	Name of the worksheet being formatted; default: `"ClusterSummary"`
`df`	data.frame containing the cluster summary from the `ConservedWaters()` function; *e.g.*, `h2o.clusters.summary`

Details

This function is to *ONLY* be used with the results of `ConservedWaters()`. Specific aspects of how the returned `data.frame` will be formatted are **hard-coded** into this function.

This openxlsx function is **NOT** exported.

Value

The workbook containing the indicated and newly formatted worksheet.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "openxlsx functions": `openxlsxCellStyles`, `oxAlignOverlapSheet`, `oxClusterStatsSheet`, `oxInitWaterDataSheet`, `oxPDBcleanedSummarySheet`, `oxPlainDataSheet`, `oxRCSBinfoSheet`, `oxWaterOccurrenceSheet`

`oxInitWaterDataSheet` *Initial Water Data Sheet*

Description

Constructs the openxlsx worksheet for the initial water data.

Usage

```
oxInitWaterDataSheet(wb.name, sheet.name = "InitialWaterData", df)
```

Arguments

`wb.name`	Name of the workbook for the results; *e.g.*, results.wb
`sheet.name`	Name of the worksheet being formatted; default: `"InitialWaterData"`
`df`	data.frame containing the concatenate initial waters with experimental and experimentally derived values obtatined within the `ConservedWaters()` function; *e.g.*, `h2o.df`

Details

This function is to *ONLY* be used with the results of `ConservedWaters()`. Specific aspects of how the returned `data.frame` will be formatted are **hard-coded** into this function.

This openxlsx function is **NOT** exported.

Value

The workbook containing the indicated and newly formatted worksheet.

Author(s)

Emilio Xavier Esposito `<emilio@exeResearch.com>`

See Also

Other "openxlsx functions": `openxlsxCellStyles`, `oxAlignOverlapSheet`, `oxClusterStatsSheet`, `oxClusterSummarySheet`, `oxPDBcleanedSummarySheet`, `oxPlainDataSheet`, `oxRCSBinfoSheet`, `oxWaterOccurrenceSheet`

`oxPDBcleanedSummarySheet` *Cleaned PDB Structures Data Sheet*

Description

Constructs the openxlsx worksheet for the `CleanProteinStructures()` results.

Usage

```
oxPDBcleanedSummarySheet(wb.name, sheet.name = "PDBcleanedSummary", df)
```

Arguments

`wb.name`	Name of the workbook for the results; *e.g.*, results.wb
`sheet.name`	Name of the worksheet being formatted; default: `"PDBcleanedSummary"`
`df`	data.frame containing the summary of `CleanProteinStructures()`; *e.g.*, `df.results`

Details

This function is to *ONLY* be used with the results of `CleanProteinStructures()`. Specific aspects of how the returned `data.frame` will be formatted are **hard-coded** into this function.

Notable formatting:

- Top row frozen
- Column widths are set based on column content
- Structures with hydrogen atoms removed are highlighted with amber cell color
- Structures with OoR values, modeled atoms, and removed waters are highlighted with amber cell color

This openxlsx function is **NOT** exported.

Value

The workbook containing the indicated and newly formatted worksheet.

Author(s)

Emilio Xavier Esposito `<emilio@exeResearch.com>`

See Also

Other "openxlsx functions": `openxlsxCellStyles`, `oxAlignOverlapSheet`, `oxClusterStatsSheet`, `oxClusterSummarySheet`, `oxInitWaterDataSheet`, `oxPlainDataSheet`, `oxRCSBinfoSheet`, `oxWaterOccurrenceSheet`

`oxPlainDataSheet` *Plain Data Sheet*

Description

Constructs a plain Excel worksheet via the openxlsx package.

Usage

```
oxPlainDataSheet(wb.name, sheet.name = "basic", df)
```

Arguments

`wb.name`	Name of the workbook for the results; *e.g.*, results.wb
`sheet.name`	Name of the worksheet being formatted; default: `"basic"`
`df`	data.frame containing the data to be written; *e.g.*, `df.results`

Details

This function creates a basic Excel worksheet with minimal formatting.

Notable formatting:

- Top row frozen
- Column widths are set based on column content

This openxlsx function is **NOT** exported.

Value

The workbook containing the indicated and newly formatted worksheet.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "openxlsx functions": `openxlsxCellStyles`, `oxAlignOverlapSheet`, `oxClusterStatsSheet`, `oxClusterSummarySheet`, `oxInitWaterDataSheet`, `oxPDBcleanedSummarySheet`, `oxRCSBinfoSheet`, `oxWaterOccurrenceSheet`

oxRCSBinfoSheet *openxlsx PDB/RCSB Summary Sheet*

Description

Constructs the openxlsx worksheet for the Similarity Summary analysis.

Usage

```
oxRCSBinfoSheet(wb.name, sheet.name = "RCSB_information", df)
```

Arguments

wb.name	Name of the workbook for the results; *e.g.*, results.wb
sheet.name	Name of the worksheet being formatted; default: "PDB_information"
df	data.frame containing the PDB/RCSB information obtatined within the ConservedWaters() function; *e.g.*, pdbs.information

Details

This function is to *ONLY* be used with the results of ConservedWaters(). Specific aspects of how the returned data.frame will be formatted are **hard-coded** into this function.

This openxlsx function is **NOT** exported.

Value

The workbook containing the indicated and newly formatted worksheet.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "openxlsx functions": openxlsxCellStyles, oxAlignOverlapSheet, oxClusterStatsSheet, oxClusterSummarySheet, oxInitWaterDataSheet, oxPDBcleanedSummarySheet, oxPlainDataSheet, oxWaterOccurrenceSheet

`oxWaterOccurrenceSheet` *openxlsx Water Occurrence Summary*

Description

Constructs the openxlsx worksheet for the Water Occurrence summary.

Usage

```
oxWaterOccurrenceSheet(wb.name, sheet.name = "WaterOccurrenceSummary",
  df)
```

Arguments

`wb.name`	Name of the workbook for the results; *e.g.*, results.wb
`sheet.name`	Name of the worksheet being formatted; default: `"WaterOccurrenceSummary"`
`df`	data.frame containing the water occurrence results of the `ConservedWaters()` function; *e.g.*, `h2o.occurrence`

Details

This function is to *ONLY* be used with the results of `ConservedWaters()`. Specific aspects of how the returned `data.frame` will be formatted are **hard-coded** into this function.

This openxlsx function is **NOT** exported.

Value

The workbook containing the indicated and newly formatted worksheet.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "openxlsx functions": `openxlsxCellStyles`, `oxAlignOverlapSheet`, `oxClusterStatsSheet`, `oxClusterSummarySheet`, `oxInitWaterDataSheet`, `oxPDBcleanedSummarySheet`, `oxPlainDataSheet`, `oxRCSBinfoSheet`

PDB.1ecd *PDB Structure of Erythrocruorin*

Description

Structure of erythrocruorin in different ligand states refined at 1.4 A resolution.

Details

The 3D structure of erythrocruorin in different ligand states refined at 1.4 A resolution. This 3D structure was downloaded from the RCSB and read into R using `bio3d::read.pdb()`. It is used in examples and testing.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

PDB ID: 1ecd Wolfgang Steigemann and Ernst Weber. Structure of erythrocruorin in different ligand states refined at 1.4 A resolution. *Journal of Molecular Biology*, 1979, **127** (*3*), pp 309-338. DOI: 10.1016/0022-2836(79)90332-2 PMID: 430568

HM Berman, J Westbrook, Z Feng, G Gilliland, TN Bhat, H Weissig, IN Shindyalov, PE Bourne. The Protein Data Bank. *Nucleic Acids Research*, 2000, **28** (*1*), pp 235-242. DOI: 10.1093/nar/28.1.235 PMCID: PMC102472

PDB.5rxn *PDB Structure of Rubredoxin*

Description

Combined crystallographic refinement and energy minimization of rubredoxin at 1.2 Angstrom resolution.

Details

The 3D structure of rubredoxin at 1.2 Angstrom resolution obtained via combined crystallographic refinement and energy minimization. This 3D structure was downloaded from the RCSB and read into R using `bio3d::read.pdb()`. It is used in examples and testing.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

PDB ID: 5rxn Keith D Watenpaugh. Combined Crystallographic Refinement And Energy Minimization Of Rubredoxin At 1.2 Angstrom Resolution.

HM Berman, J Westbrook, Z Feng, G Gilliland, TN Bhat, H Weissig, IN Shindyalov, PE Bourne. The Protein Data Bank. *Nucleic Acids Research*, 2000, **28** (*1*), pp 235-242. DOI: 10.1093/nar/28.1.235 PMCID: PMC102472

`ProtHetWatIndices` *Protein, HET, and Water Atom Indices*

Description

Indices for the protein, HET-atom, and water atoms

Usage

```
ProtHetWatIndices(data)
```

Arguments

`data`	The atom data.frame of the PDB read into the `R` session using the function `bio3d::read.pdb()`.

Details

Returns individual numerical vectors for the protein, HET-atom, and water atoms from the atom `base::data.frame()` of a PDB.

NOTE: This is a non-public function.

Value

Individual vectors for the indices of the protein, HET-atom, and water atoms for a PDB file.

Author(s)

Emilio Xavier Esposito `<emilio@exeResearch.com>`

See Also

Other utilities: `ConservationSet`, `DetermineChainsOfInterest`, `ExtractFileTimeStamp`, `ExtractPDBids`, `FileTimeStamp`, `HasXWaters`, `Nearby`, `RescaleValues`, `RetainChainsOfInterest`, `ReturnPDBfullPath`, `StandardizeAsparticAcidNames`, `StandardizeCysteineNames`, `StandardizeGlutamicAcidNames`, `StandardizeHistidineNames`, `StandardizeLysineNames`, `TimeSpan`, `UniqueAtomHashes`, `aaStandardizeNames`, `getAtomTypeCounts`, `getResTypeCounts`, `res2xyz`, `resAtomType2AtomClass`, `write.basic.pdb`, `write.conservedWaters.pdb`

Examples

```
## Not run:
ProtHetWatIndices(thrombin.1hai$atom[c(1:10, 2341:2350, 2385:2394), ])
# $prot.idc
# [1]  1  2  3  4  5  6  7  8  9 10
#
# $het.idc
# [1] 11 12 13 14 15 16 17 18 19 20
#
# $h2o.idc
# [1] 21 22 23 24 25 26 27 28 29 30

## End(Not run)
```

RemoveHydrogenAtoms *Remove Hydrogen and Deuterium Atoms*

Description

Removes hydrogen atoms from a RCSB/PDB structure.

Usage

```
RemoveHydrogenAtoms(atoms.chains.oi)
```

Arguments

`atoms.chains.oi`	The data.frame containing the PDB file information; aka the PDB structure

Details

Removes hydrogen and deuterium atoms from a PDB formatted `base::data.frame()` with PDB formatted information.

Value

data.frame of the PDB structure *without* hydrogen or deuterium atoms

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Clean Protein Structure": `CleanProteinStructures`, `RemoveModeledAtoms`, `RemoveOoR.b`, `RemoveOoR.o`, `RetainWatersWithinX`

Examples

```
PDB.5rxn.noHydrogens <- RemoveHydrogenAtoms(PDB.5rxn$atom)
```

`RemoveModeledAtoms` *Remove Modeled Atoms*

Description

Removes modeled atoms from a RCSB/PDB structure.

Usage

```
RemoveModeledAtoms(atoms.chains.oi)
```

Arguments

`atoms.chains.oi`
: The data.frame containing the PDB file information; aka the PDB structure

Details

Sometimes atoms are not well resolved within the electron density maps and the scientists resolving/determining the structures "model back into" the resulting structure the atoms based on historical data. This is most common for residues where a portion of the residue is missing and based on the structure the missing atoms are replaces. These modeled atoms have an occupancy value of 0.01 or less and are identified and removed.

The reported occupancy value of 0.01 is used as the cutoff because several PDB structures have comments in the `REMARK 3` section stating, "`...MISSING ABOVE 1SIGMA WERE GIVEN A 0.01 OCCUPANCY...`"`` or "...WITH NO DENSITIES ARE GIVEN OCCUPANCY VALUES OF 0.01..."`.

Value

data.frame of the PDB structure *without* the modeled atoms

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Clean Protein Structure": `CleanProteinStructures`, `RemoveHydrogenAtoms`, `RemoveOoR.b`, `RemoveOoR.o`, `RetainWatersWithinX`

Examples

```
PDB.1ecd.noModeledAtoms <- RemoveModeledAtoms(PDB.1ecd$atom)
```

RemoveOoR.b *Remove B-value Out of Range Atoms*

Description

Removes atoms with B-values out of accepted range.

Usage

```
RemoveOoR.b(atoms.chains.oi)
```

Arguments

atoms.chains.oi	The data.frame containing the PDB file information; aka the PDB structure

Details

Accepted B-value values range from 0 to 100 with values. Atoms are considered stationary – possessing low thermal energy – when possessing values between 20 and 40 while larger values between 60 and 100 indicate a large amount of position variability within the lattice. This function identifies occupancy values less than 0 and greater than 100 and removes them from the structure.

Value

data.frame of the PDB structure *without* the offending atoms

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Clean Protein Structure": CleanProteinStructures, RemoveHydrogenAtoms, RemoveModeledAtoms, RemoveOoR.o, RetainWatersWithinX

Examples

```
nrow(PDB.4ape$atom)
PDB.4ape.OoR.b <- RemoveOoR.b(PDB.4ape$atom)
nrow(PDB.4ape.OoR.b)
```

RemoveOoR.o *Remove Occupancy Out of Range Atoms*

Description

Removes atoms with occupancy values out of accepted range.

Usage

```
RemoveOoR.o(atoms.chains.oi)
```

Arguments

atoms.chains.oi	The data.frame containing the PDB file information; aka the PDB structure

Details

Accepted occupancy values range from 0 to 1 with values for modeled atoms being 0.0 or 0.01 and highly conserved or represented atoms throughout the lattice having values greater than 0.9 and commonly possessing values of 1.0. This function identifies occupancy values less than 0 and greater than 1 and removes them from the structure.

Value

data.frame of the PDB structure *without* the offending atoms

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Clean Protein Structure": CleanProteinStructures, RemoveHydrogenAtoms, RemoveModeledAtoms, RemoveOoR.b, RetainWatersWithinX

Examples

```
nrow(PDB.4dfr$atom)
PDB.4dfr.OoR.o <- RemoveOoR.o(PDB.4dfr$atom)
nrow(PDB.4dfr.OoR.o)
```

res2xyz *Residue Indices to Coordinate Indices*

Description

Return the coordinate indices for the provided residue indices.

Usage

```
res2xyz(res.idc)
```

Arguments

res.idc	Indicies of residues to convert to coordinate indices

Details

Using the residue indices of the atoms base::data.frame() (*e.g.*, pdb$atom) determine the coodinate indices of the residue atoms (*e.g.*, pdb$xyz).

Value

Vector of coordinate indicies to be applied to pdb$xyz

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
res.idc <- c(5:10)
res2xyz(res.idc)
# [1] 13 15 15 16 18 18 19 21 21 22 24 24 25 27 27 28 30
```

`resAtomType2AtomClass` *Convert Residue-AtomType to AtomType Class*

Description

Converts the residue-AtomType to AtomType Class.

Usage

```
resAtomType2AtomClass(resAT)
```

Arguments

`resAT`	residue and AtomType; *e.g.*, `"LYS NZ"`

Details

See examples...

Value

A string with the AtomType's class:

- Nitrogen
- Nitrogen (+)
- Oxygen
- Oxygen (-)
- Carbon
- Sulfur

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: `ConservationSet`, `DetermineChainsOfInterest`, `ExtractFileTimeStamp`, `ExtractPDBids`, `FileTimeStamp`, `HasXWaters`, `Nearby`, `ProtHetWatIndices`, `RescaleValues`, `RetainChainsOfInterest`, `ReturnPDBfullPath`, `StandardizeAsparticAcidNames`, `StandardizeCysteineNames`, `StandardizeGlutamicAcidNames`, `StandardizeHistidineNames`, `StandardizeLysineNames`, `TimeSpan`, `UniqueAtomHashes`, `aaStandardizeNames`, `getAtomTypeCounts`, `getResTypeCounts`, `res2xyz`, `write.basic.pdb`, `write.conservedWaters.pdb`

Examples

```
resAtomType2AtomClass(resAT="LYS NZ")
# [1] "Nitrogen (+)"
resAtomType2AtomClass(resAT="GLU N")
# [1] "Nitrogen"
resAtomType2AtomClass(resAT="VAL O")
# [1] "Oxygen"
resAtomType2AtomClass(resAT="ASP OD2")
# [1] "Oxygen (-)"
resAtomType2AtomClass(resAT="GLN CA")
# [1] "Carbon"
resAtomType2AtomClass(resAT="CYS SG")
# [1] "Sulfur"
```

RescaleValues *Rescale Values*

Description

Rescales provided vector of values to a user defined range.

Usage

```
RescaleValues(data, newMin = 0, newMax = 1)
```

Arguments

data	A vector of numerical values to be rescaled
newMin	A numerical value indicating the new minimum value; default: 0
newMax	A numerical value indicating the new maximum value; default: 1

Details

Rescale the values to a new user defined range.

Value

vector of rescaled numerical values.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
RescaleValues(0:10, newMin = 0, newMax = 1)
# [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

RetainChainsOfInterest

Retain Chains Of Interest

Description

Retain chains of interest based on user input parameters

Usage

```
RetainChainsOfInterest(atoms.oi, chains.explore, chains.oi)
```

Arguments

atoms.oi	data.frame containing the PDB of the protein
chains.explore	String of chains to explore
chains.oi	String from the DetermineChainsOfInterest() function indicating if "first", "all", or a "user" defined set of chains should be used. **NOTE**: "first" is alphabetically first. Thus if the order within the original PDB file is L and then H, this function will return H because it is alphebetically first.

Details

Using the user provided chains of interest, indicate the PDB chains to retain.

NOTE: This is a **non-public** function and is **NOT** available for general use. Please contact the author if you believe this function should be available for general use.

Value

data.frame of the protein atoms retained based on the indicated chains of interest

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
  ## Not run:
  RetainChainsOfInterest(PDB.4dfr$atom, "B", "user")
  RetainChainsOfInterest(PDB.1hai$atom, c("H", "L"), "user")
  RetainChainsOfInterest(PDB.4dfr$atom, "A", "first")

## End(Not run)
```

RetainWatersWithinX *Retain Waters Within X Angstroms of Protein*

Description

Retains water oxygen atoms within a user defined distance

Usage

```
RetainWatersWithinX(atoms.dist, prot.het.h2o.idc, cutoff.prot.h2o.dist)
```

Arguments

atoms.dist	Atomic distances calculated with the stats::dist() function
prot.het.h2o.idc	List of protein, HET-atom, and water atom indices
cutoff.prot.h2o.dist	User defined maximum numerical distance, in Angstroms, between the protein and water oxygen atoms to be retained.

Details

Retain water oxygen atoms within a user defined distance. This function is a coarse grain method of removing waters beyond a predefined distance to reduce the computational load associated with the stats::dist() function for a collection of protein structure.

Value

numerical vector of water oxygen atom indicies to retain

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other "Clean Protein Structure": CleanProteinStructures, RemoveHydrogenAtoms, RemoveModeledAtoms, RemoveOoR.b, RemoveOoR.o

Examples

```
##--- determine the protein, hetatom, and  water indices
prot.het.h2o.idc <- ProtHetWatIndices(data=PDB.1hah.aoi.clean)

##--- calculate the distances
atoms.dist <- as.matrix(dist(PDB.1hah.aoi.clean[, c("x","y","z")],
                             method="euclidean",
                             diag=TRUE, upper=TRUE))
diag(atoms.dist) <- NA

water.idc.within.6 <- RetainWatersWithinX(atoms.dist,
                                          prot.het.h2o.idc,
                                          cutoff.prot.h2o.dist=6.0)
# - 204 of the 204 water oxygen atoms are within 6 Angstroms of the protein
```

ReturnPDBfullPath *Return PDB Full Path*

Description

Determine the full path of the PDB files and return the complete path of each file within the provided directory.

Usage

```
ReturnPDBfullPath(prefix)
```

Arguments

prefix	The directory with the PDB files of interest; *e.g.*, ProteinSystem_Aligned

Details

The complete path of the PDB file(s) in the user provided `prefix` is returned.

NOTE: This is a non-public function.

Value

collection of string values with the complete (normalized) path for each PDB file within the provided directory/folder.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: `ConservationSet`, `DetermineChainsOfInterest`, `ExtractFileTimeStamp`, `ExtractPDBids`, `FileTimeStamp`, `HasXWaters`, `Nearby`, `ProtHetWatIndices`, `RescaleValues`, `RetainChainsOfInterest`, `StandardizeAsparticAcidNames`, `StandardizeCysteineNames`, `StandardizeGlutamicAcidNames`, `StandardizeHistidineNames`, `StandardizeLysineNames`, `TimeSpan`, `UniqueAtomHashes`, `aaStandardizeNames`, `getAtomTypeCounts`, `getResTypeCounts`, `res2xyz`, `resAtomType2AtomClass`, `write.basic.pdb`, `write.conservedWaters.pdb`

`StandardizeAsparticAcidNames` *Standardize Aspartic Acid Names*

Description

Standardize the protonated aspartic acid three-letter residue name to ASP.

Usage

```
StandardizeAsparticAcidNames(residue.names)
```

Arguments

`residue.names`	A vector of strings containing the three-letter residue names (strings)

Details

The the protonated aspartic acid three-letter residue name (ASH) is converted to the standard "ASP" residue name. This function is part of the `aaStandardizeNames()`.

NOTE: This is a non-public function.

Value

vector of three-letter residue names with *standardized* aspartic acid residue names

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeCysteineNames, StandardizeGlutamicAcidNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
  ## Not run:
  residue.names <- c("HIS", "HID", "HIE", "HIP", "HSD", "HSE", "HSP",
                     "CYS", "CYM", "CYX", "ASP", "ASH", "GLU", "GLH",
                     "LYS", "LYN")
  StandardizeAsparticAcidNames(residue.names)
  # [1] "HIS" "HID" "HIE" "HIP" "HSD" "HSE" "HSP" "CYS" "CYM" "CYX"
  #     "ASP" "ASP" "GLU" "GLH" "LYS" "LYN"

## End(Not run)
```

StandardizeCysteineNames
Standardize Cysteine Names

Description

Standardize the two three-letter cysteine residue names to CYS.

Usage

```
StandardizeCysteineNames(residue.names)
```

Arguments

residue.names	A vector of strings containing the three-letter residue names (strings)

Details

The two three-letter cysteine residue names used to indicate the different cystine states (CYM: deprotonated cysteine and CYX: no proton, neutral charge, part of a disulfide bridge) are converted to the standard "CYS" (protonated) residue name. This function is part of the aaStandardizeNames().

NOTE: This is a non-public function.

Value

vector of three-letter residue names with *standardized* cysteine residue names

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeGlutamicAcidNames, StandardizeHistidineName StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
  ## Not run:
  residue.names <- c("HIS", "HID", "HIE", "HIP", "HSD", "HSE", "HSP",
                     "CYS", "CYM", "CYX", "ASP", "ASH", "GLU", "GLH",
                     "LYS", "LYN")
  StandardizeCysteineNames(residue.names)
  # [1] "HIS" "HID" "HIE" "HIP" "HSD" "HSE" "HSP" "CYS" "CYS" "CYS"
  #     "ASP" "ASH" "GLU" "GLH" "LYS" "LYN"

## End(Not run)
```

StandardizeGlutamicAcidNames

Standardize Glutamic Acid Names

Description

Standardize the protonated glutamic acid three-letter residue name to GLU.

Usage

```
StandardizeGlutamicAcidNames(residue.names)
```

Arguments

residue.names	A vector of strings containing the three-letter residue names (strings)

Details

The the protonated glutamic acid three-letter residue name (GLH) is converted to the standard "GLU" residue name. This function is part of the aaStandardizeNames().

NOTE: This is a non-public function.

Value

vector of three-letter residue names with *standardized* glutamic acid residue names

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
  ## Not run:
  residue.names <- c("HIS", "HID", "HIE", "HIP", "HSD", "HSE", "HSP",
                     "CYS", "CYM", "CYX", "ASP", "ASH", "GLU", "GLH",
                     "LYS", "LYN")
  StandardizeGlutamicAcidNames(residue.names)
  # [1] "HIS" "HID" "HIE" "HIP" "HSD" "HSE" "HSP" "CYS" "CYM" "CYX"
  #     "ASP" "ASH" "GLU" "GLU" "LYS" "LYN"

## End(Not run)
```

StandardizeHistidineNames

Standardize Histidine Names

Description

Standardize the various three-letter histine residue names to HIS.

Usage

```
StandardizeHistidineNames(residue.names)
```

Arguments

residue.names	A vector of strings containing the three-letter residue names (strings)

Details

The various three-letter histidine residue names ("HID", "HIE", "HIP", "HSD", "HSE", "HSP") used to indicate the different protonation states are converted to the standard "HIS" residue name. This function is part of the aaStandardizeNames().

NOTE: This is a non-public function.

Value

vector of three-letter residue names with *standardized* histidine residue names

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
  ## Not run:
  residue.names <- c("HIS", "HID", "HIE", "HIP", "HSD", "HSE", "HSP",
                     "CYS", "CYM", "CYX", "ASP", "ASH", "GLU", "GLH",
                     "LYS", "LYN")
  StandardizeHistidineNames(residue.names)
  # [1] "HIS" "HIS" "HIS" "HIS" "HIS" "HIS" "HIS" "CYS" "CYM" "CYX"
  #     "ASP" "ASH" "GLU" "GLH" "LYS" "LYN"

## End(Not run)
```

StandardizeLysineNames

Standardize Lysine Names

Description

Standardize the de-protonated lysine three-letter residue name to LYS.

Usage

```
StandardizeLysineNames(residue.names)
```

Arguments

residue.names	A vector of strings containing the three-letter residue names (strings)

Details

The the de-protonated lysine three-letter residue name (LYN) is converted to the standard "LYS" residue name. This function is part of the aaStandardizeNames().

NOTE: This is a non-public function.

Value

vector of three-letter residue names with *standardized* lysine residue names

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames StandardizeHistidineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
  ## Not run:
  residue.names <- c("HIS", "HID", "HIE", "HIP", "HSD", "HSE", "HSP",
                     "CYS", "CYM", "CYX", "ASP", "ASH", "GLU", "GLH",
                     "LYS", "LYN")
  StandardizeLysineNames(residue.names)
  # [1] "HIS" "HID" "HIE" "HIP" "HSD" "HSE" "HSP" "CYS" "CYM" "CYX"
  #     "ASP" "ASH" "GLU" "GLH" "LYS" "LYS"

## End(Not run)
```

---

thrombin.1hai *PDB Structure of Thrombin*

Description

Isomorphous structures of prethrombin2, hirugen-, and PPACK-thrombin.

Details

The 3D structure of isomorphous structures of prethrombin2, hirugen-, and PPACK-thrombin at 2.4 Angstroms. This 3D structure was downloaded from the RCSB and read into R using bio3d::read.pdb(). It is used in examples and testing.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

References

PDB ID: 1hai J Vijayalakshmi, KP Padmanabhan, KG Mann, and A Tulinsky. The isomorphous structures of prethrombin2, hirugen-, and PPACK-thrombin: changes accompanying activation and exosite binding to thrombin.. *Protein Science*, 1994, **3** (*12*), pp 2254–2271.

HM Berman, J Westbrook, Z Feng, G Gilliland, TN Bhat, H Weissig, IN Shindyalov, PE Bourne. The Protein Data Bank. *Nucleic Acids Research*, 2000, **28** (*1*), pp 235-242. DOI: 10.1002/pro.5560031211 PMCID: PMC2142772

`thrombin10.PDBs.align` *Thrombin10 Vignette's Primary Sequence Alignment*

Description

Thrombin10 vignette's primary sequence alignment.

Details

The primary sequence alignment is being provided because the CRAN servers where R packages are tested does not have the MUltiple Sequence Comparison by Log-Expectation (MUSCLE; MUSCLE webpage and the EBI webpage application installed. This alignment of 10 thrombin structures for the Thrombin 10 Vignette allows the vignette to be completed without errors.

Author(s)

Emilio Xavier Esposito `<emilio@exeResearch.com>`

References

RC Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 2004, **32** (*5*), pp 1792-1797.
PMID: 15034147
PMCID: PMC390337
DOI: 10.1093/nar/gkh340

RC Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity *BMC Bioinformatics*, 2004, 5:113.
PMID: 15318951
PMCID: PMC517706
DOI: 10.1186/1471-2105-5-113

TimeSpan *Time Span*

Description

Calculate the duration of a set of calculations.

Usage

```
TimeSpan(time.start)
```

Arguments

time.start	The start time determined using the base::Sys.time()

Details

Using the time a set of calculations started, the duration of the calculations is returned.

NOTE: This is a non-public function.

Value

character string of the calculation duration

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames StandardizeHistidineNames, StandardizeLysineNames, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb, write.conservedWaters.pdb

Examples

```
  ## Not run:
  time.start <- Sys.time() - 25   ## subtract 25 seconds from time.start
  TimeSpan(time.start)
  # [1] "00:00:25"

## End(Not run)
```

`UniqueAtomHashes` *Create Unique Atom Hashes*

Description

Constructs unique atom hashes from the provided

Usage

```
UniqueAtomHashes(atoms.oi, cols.oi, separator = "_")
```

Arguments

`atoms.oi`	A data.frame containing the common PDB information in columns
`cols.oi`	A vector of column names to be used in the construction of the unique atom hashes
`separator`	A single character string to separate the atom specific identifiers. Acceptable separators include: _ (default), -, +, ., :, |, " " (space), and "" (no separator).

Details

Using atom specific identifiers from a PDB-like formatted data.frame, unique atom hashes are constructed. The identifiers are separated by a user-defined separator, the default separator is an underscores ("_"), and the constructed hashes are returned as a vector.

Select a separator to allow easy splitting of the the unique atom hashes using the `base::strsplit()` function to access the individual components.

NOTE: This is a non-public function.

Value

a vector of strings containing the unique atom hashes

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: `ConservationSet`, `DetermineChainsOfInterest`, `ExtractFileTimeStamp`, `ExtractPDBids`, `FileTimeStamp`, `HasXWaters`, `Nearby`, `ProtHetWatIndices`, `RescaleValues`, `RetainChainsOfInterest`, `ReturnPDBfullPath`, `StandardizeAsparticAcidNames`, `StandardizeCysteineNames`, `StandardizeGlutamicAcidNames`, `StandardizeHistidineNames`, `StandardizeLysineNames`, `TimeSpan`, `aaStandardizeNames`, `getAtomTypeCounts`, `getResTypeCounts`, `res2xyz`, `resAtomType2AtomClass`, `write.basic.pdb`, `write.conservedWaters.pdb`

Examples

```
## Not run:
atoms.oi <- thrombin.1hai$atom[1:10, ]
cols.oi <- c("elety", "resid", "chain", "resno")
UniqueAtomHashes(atoms.oi, cols.oi, separator = "_")
# [1] "N_THR_L_1"   "CA_THR_L_1"  "C_THR_L_1"   "O_THR_L_1"   "CB_THR_L_1"
#     "OG1_THR_L_1" "CG2_THR_L_1" "N_PHE_L_1"   "CA_PHE_L_1"  "C_PHE_L_1"

UniqueAtomHashes(atoms.oi, cols.oi, separator = "!")
# The provided separator "!" is not acceptable. The default separator "_" is being used.
#  [1] "N_THR_L_1"   "CA_THR_L_1"  "C_THR_L_1"   "O_THR_L_1"   "CB_THR_L_1"
#      "OG1_THR_L_1" "CG2_THR_L_1" "N_PHE_L_1"   "CA_PHE_L_1"  "C_PHE_L_1"

## End(Not run)
```

---

vanddraabe *vanddraabe: Identification and Statistical Analysis of Conserved Waters in Proteins*

Description

Identify and analyze conserved waters within crystallographic protein structures and molecular dynamics simulation trajectories. Statistical parameters for each water cluster, informative graphs, and a PyMOL session file to visually explore the conserved waters and protein are returned. Hydrophilicity is the propensity of waters to congregate near specific protein atoms and is related to conserved waters. An informatics derived set of hydrophilicity values are provided based on a large, high-quality X-ray protein structure dataset.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

---

write.basic.pdb *Write Basic PDB File*

Description

Writes standard PDB file.

Usage

```
write.basic.pdb(file, atoms.oi)
```

Arguments

file	Filename with ".pdb" extension.
atoms.oi	The atoms base::data.frame().

Details

Using the bio3d::write.pdb() function this function writes a PDB file from a base::data.frame() containing the typical PDB file information. This function is called from the FreeSASA.diff() function within the HydrophilicityEvaluation() function.

NOTE: This is a non-public function.

Value

Writes a PDB file for the FreeSASA.diff() function.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.conservedWaters.pdb

Examples

```
  ## Not run:
    write.basic.pdb(file = "just_some_PDB.pdb", atoms.oi)

## End(Not run)
```

---

write.conservedWaters.pdb

Write Conserved Waters to PDB File

Description

Writes conserved water information to a PDB file.

Usage

```
write.conservedWaters.pdb(file, h2o.clusters.summary)
```

Arguments

file	Filename with ".pdb" extension.
h2o.clusters.summary	The conserved water clusters summary.

Details

Using the bio3d::write.pdb() function this function writes a PDB file for the conserved water oxygen atoms with the percentage of structures with a water participting in the cluster (written to the occupancy column) and the calculated B-value – using the rmsf of the waters in the cluster – for the waters participating in the cluster (written to the B-value column). This function is called from the ConservedWaters() function.

All water molecules will include the water's oxygen atom (elety), be assigned the residue name (resid) HOH, and the chain (chain) A while the atom number (eleno) and residue number (resno) both start at 1.

NOTE: This is a non-public function.

Value

Writes a PDB file with the X, Y, and Z coordinates, percent conserved within the analyzed structures, and the calculated B-value for the oxygen atoms of the clustered waters.

Author(s)

Emilio Xavier Esposito <emilio@exeResearch.com>

See Also

Other utilities: ConservationSet, DetermineChainsOfInterest, ExtractFileTimeStamp, ExtractPDBids, FileTimeStamp, HasXWaters, Nearby, ProtHetWatIndices, RescaleValues, RetainChainsOfInterest, ReturnPDBfullPath, StandardizeAsparticAcidNames, StandardizeCysteineNames, StandardizeGlutamicAcidNames, StandardizeHistidineNames, StandardizeLysineNames, TimeSpan, UniqueAtomHashes, aaStandardizeNames, getAtomTypeCounts, getResTypeCounts, res2xyz, resAtomType2AtomClass, write.basic.pdb

Examples

```
## Not run:
  write.conservedWaters.pdb(file = "system_conservedWaters.pdb",
                            h2o.clusters.summary)

## End(Not run)
```

Index