Package 'SpacePAC'

November 12, 2024

Type Package

Title Identification of Mutational Clusters in 3D Protein Space via Simulation.

Version 1.44.0

Date 2018-08-20

Author Gregory Ryslik, Hongyu Zhao

Maintainer Gregory Ryslik <gregory.ryslik@yale.edu>

Description Identifies clustering of somatic mutations in proteins via a simulation approach while considering the protein's tertiary structure.

biocViews Clustering, Proteomics

License GPL-2

Depends R(>= 2.15),iPAC

Suggests RUnit, BiocGenerics, rgl

git_url https://git.bioconductor.org/packages/SpacePAC

git_branch RELEASE_3_20

git_last_commit f6b8076

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-11

Contents

SpacePAC-package *Identifying mutational clusters in 3D protein space using simulation.*

Description

The *SpacePAC* package identifies non-random amino acid clusters in proteins in 3D space and is a sister package to *iPAC* and *GraphPAC*. *SpacePAC* considers 1, 2 or 3 non-overlapping spheres with radii specified by the user and through simulation, attempts to identify spheres where there are more mutations than expected by random chance alone. These results are then outputted in the form of a list with p-values.

Details

Please see `get.Positions` and `get.AlignedPositions` in the **iPAC** package for information about obtaining positional data.

Author(s)

Gregory Ryslik, Yuwei Cheng, Hongyu Zhao

Maintainer: Gregory Ryslik <gregory.ryslik@yale.edu>

References

Gregory Ryslik and Hongyu Zhao (2012). iPAC: Identification of Protein Amino acid Clustering. R package version 1.1.3. http://www.bioconductor.org/.

Gregory Ryslik and Hongyu Zhao (2013). GraphPAC: Identification of Mutational Clusters in Proteins via a Graph Theoretical Approach. R Package version 1.0.0 http://www.bioconductor.org/.

Bioconductor: Open software development for computational biology and bioinformatics R. Gentleman, V. J. Carey, D. M. Bates, B.Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, and others 2004, Genome Biology, Vol. 5, R80

See Also

`get.Positions` `SpaceClust`

Examples

```
## Not run:
CIF <- "https://files.rcsb.org/view/3GFT.cif"
Fasta <- "https://www.uniprot.org/uniprot/P01116-2.fasta"
KRAS.Positions <- get.Positions(CIF, Fasta, "A")
data(KRAS.Mutations)

#Calculate the required clusters
SpaceClust(KRAS.Mutations, KRAS.Positions$Positions, radii.vector = c(1,2,3,4))


## End(Not run)
```

`make.3D.Sphere` *Plots a sphere centered at the specified amino acid with a specified radius.*

Description

Plots a sphere of radius r centered at a specific residue. Currently only 1 sphere can be plotted. The rgl pckage is required.

Usage

```
make.3D.Sphere(position.matrix, center, radius, alpha = 0.5)
```

Arguments

`position.matrix`	A dataframe consisting of six columns: 1) Residue Name, 2) Amino Acid number in the protein, 3) Side Chain, 4) X-coordinate, 5) Y-coordinate and 6) Z-coordinate. Please see `get.Positions` and `get.AlignedPositions` in the **iPAC** package.
`center`	The residue number you want the sphere centered at. Use the number from the "Can.Count" column in the position matrix.
`radius`	The radius of the sphere you would like to draw.
`alpha`	The "darkness" of the sphere.

Value

The rgl package is called and a graph is shown.

Note

This function is made for ease of use in preliminary analysis only. For more sophisticated graphing packages, consider using PyMol at: http://www.pymol.org

References

Daniel Adler and Duncan Murdoch (2013). rgl: 3D visualization device system (OpenGL). R package version 0.93.935. http://CRAN.R-project.org/package=rgl

See Also

`get.Positions`

Examples

```
## Not run:
#Plots a sphere centered around amino acid 12 with radius 3.
library(rgl)

#loads the data
CIF <- "https://files.rcsb.org/view/3GFT.cif"
Fasta <- "https://www.uniprot.org/uniprot/P01116-2.fasta"
```

```
KRAS.Positions <- get.Positions(CIF, Fasta, "A")

#generates the plot
make.3D.Sphere(KRAS.Positions$Positions, 12, 3)

## End(Not run)
```

SpaceClust *SpaceClust*

Description

Finds mutational clusters via simulation. There are two options currently avaiable. The first is "SimMax" and the second is "Poisson". The Poisson method is faster and finds the 1 sphere with the largest number of mutations at each radius. A bonferroni adjustment is then used to account for multiple radii. The SimMax method uses the "simMaxspheres" parameter to find the 1, 2 or 3 (non-overlapping) spheres that together have the most number of mutations. A simulation approach is then used to find the most significant clusters. Please see the vignette for further details.

Usage

```
SpaceClust(mutation.data, position.matrix, method = "SimMax", numsims = 1000,
   simMaxSpheres = 3, radii.vector, multcomp = "bonferroni", alpha = 0.05)
```

Arguments

`mutation.data`	A matrix of 0's (no mutation) and 1's (mutation) where each column represents an amino acid in the protein and each row represents an individual sample (test subject, cell line, etc). Thus if row i in column j had a 1, that would mean that the jth amino acid for person i had a nonsynonomous mutation. Please note that getting the mutation matrix is the responsibility of the user. Further, the column names of the matrix must be in the format V1, V2, ..., Vn where n is the total number of residues. One source of information is the COSMIC database http://cancer.sanger.ac.uk/cancergenome/projects/cosmic/. However, extraction from this (or any other database) is not trivial and often requires preprocessing work. In the case of COSMIC, a local SQL server must be set up and query of the user's design must be run to pull the correct mutational data. This data must then be manipulated by the user into the matrix described. Please note, that the mutational data should come from a whole gene or a whole genome study and can not be selectively chosen as that will violate the uniformity assumption that the algorithm is based on.
`position.matrix`	A dataframe consisting of six columns: 1) Residue Name, 2) Amino Acid number in the protein, 3) Side Chain, 4) X-coordinate, 5) Y-coordinate and 6) Z-coordinate. Please see `get.Positions` and `get.AlignedPositions` in the **iPAC** package.
`method`	Either "SimMax" or "Poisson". Please see the vignette for further details on the difference.
`numsims`	The number of times to simulate the mutations on the protein. For each simulation, the mutations are uniformly distributed on the protein.

`simMaxSpheres` The maximum number of spheres to consider. Currently, the implementation allows for simMaxspheres to be either 1, 2 or 3.

`radii.vector` A vector of radii. For each radius, we find the best sphere combination. If the positional data is obtained from the pdb, the radii are measured in Angstroms as the x,y,z coordinates in the PDB are in Angstroms. Thus, for instance, a radius of "5" means that all residues with their carbon-alpha atom within 5 Angstroms of the center are included in the sphere.

`multcomp` If the Poisson method is used, a multiple comparison adjustment is required to account for the multiple sphere. As the sphere iterates through the protein (centered at each amino acid), a p-value is calculated for each sphere. Options are: "Bonferroni", "BH", or "none". The "BH" method stands for the Benjamini-Hochberg FDR correction. Please see `p.adjust` for a full description.

`alpha` If the Poisson method is used, alpha is used as the cutoff value after the appropriate multiple comparison adjustment.

Details

For the SimMax method, no multiple comparison is required for different radii sizes and sphere positions. See the vignette for more information. Furthermore, note that on average, residues are 3 Angstroms apart.

Value

If the method is Poisson, the result is a list with the following components:

`result.poisson` A data frame of the most significant clusters. The data frame has the following columns for each cluster (clusters shown as rows): Center: The amino acid at which the sphere is centered. Start: The smallest numbered residue in the sphere. End: The largest numbered residue in the sphere. Positions: The mutated positions in the sphere. MutsCount: The total number of mutations in the sphere. P.Value: The p-value for the cluster. Within.Range: The residues within the sphere. Line.Length: End-Start.

`best.radii` The radii at which the lowest p-value for the most significant cluster was found. Only the matrix for this p-value is shown.

If the method is SimMax, the result is a list with the following components.

`p.value` The smallest p.value when considering 1,2 or 3 spheres. Will match the p-value for the optimal sphere configuration.

`optimal.num.spheres` The number of spheres with the most statistically significant p-value.

`optimal.radius` The radius at which the most statistically significant p-value is identified.

`optimal.sphere` This presents the sphere results with the most statistically significant p-value. It will automatically display whether 1, 2 or 3 spheres is best. In the very unlikely event that more than one sphere result has the same z-score (for instance the z-score is the same whether you consider 2 or 3 spheres), the result that uses the minimum number of spheres will be displayed.

`best.1.sphere` This shows the orientation of the most statistically significant sphere. It will display the following items: 1) Center: The amino acid at which the sphere is centered. 2) Start: The smallest numbered residue in the sphere. 3) End: The largest numbered residue in the sphere. 4) Positions: The mutated positions in

the sphere. 5) MutsCount: The total number of mutations in the sphere. 6) Z-Score: The normalized z-score as defined in the vignette. 7) Within.Range: The residues within the sphere. 8) Line.Length: End-Start.

`best.2.sphere` This shows the orientation of the most statistically significant 2 spheres. The entries are the same as the items for “best.1.sphere" except for a "1" or "2" appended to each column name. A “1" means that the information presented in the column belongs to the first sphere while a “2" means that the information in the column belongs to the second sphere. The “MutsCountTotal" column shows how many mutations are in both spheres and is just the sum of “MutsCount1" and “MutsCount2". Finally, the “Intersection" column is the intersection of “Within.Range1" and “Within.Range2" and should be blank unless an error occurs.

`best.3.sphere` This shows the orientation of the most statistically significant 3 spheres. The entries are the same as in “best2.sphere" except now there is a “1", “2" or “3" appended to each column to signify whether the 1st, 2nd or 3rd sphere is being considered.

`best.1.sphere.radius`
The radius that provides the most statistically significant result when only 1 sphere is considered.

`best.2.sphere.radius`
The radius that provides the most statistically significant result when only 2 spheres are considered.

`best.3.sphere.radius`
The radius that provides the most statistically signifificant result when only 3 spheres are considered.

`bad.2.sphere.message`
If finding the optimal 2 spheres caused an error (possibly because no non-overlapping spheres or all the mutations are on one residue) a message is shown here with more details.

`bad.3sphere.message`
If finding the optimal 3 spheres caused an error (possibly because no non-overlapping spheres or all the mutations are on one or two residues) a message is shown here with more details.

`bad.2sphere.radii`
If finding the optimal 2 spheres caused an error, the radii at which errors occurred are displayed.

`bad.3sphere.radii`
If finding the optimal 3 spheres caused an error, the radii at which errors occurred are displayed.

Note

See the ’multcomp’ package on CRAN for a description of how the multiple comparison adjustment is made.

If you use the Poisson method, a Bonferroni correction is used to adjust for all the radii. As an example, supposing that the most significant cluster is found at radius 5, and the radii vector was (1,2,3,4,5), the p-values displayed in the result matrix would be the p-value_b*5 where p-value_b is the p-value if algorithm was run with radii vector= c(5).

References

Torsten Hothorn, Frank Bretz and Peter Westfall (2008). Simultaneous Inference in General Parametric Models. Biometrical Journal 50(3), 346–363.

See Also

get.Positions SpaceClust

Examples

```
CIF <- "https://files.rcsb.org/view/3GFT.cif"
Fasta <- "https://www.uniprot.org/uniprot/P01116-2.fasta"
KRAS.Positions <- get.Positions(CIF, Fasta, "A")
data(KRAS.Mutations)


#Calculate the required clusters using SimMax
SpaceClust(KRAS.Mutations, KRAS.Positions$Positions, radii.vector = c(1,2,3,4))

#Calculate the required clusters using Poisson
SpaceClust(KRAS.Mutations, KRAS.Positions$Positions, radii.vector = c(1,2,3,4), method = "Poisson")
```

Index