Package 'iPAC'

October 18, 2024

Type Package

Title Identification of Protein Amino acid Clustering

Version 1.49.0

Date 2024-04-24

Author Gregory Ryslik, Hongyu Zhao

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

Description iPAC is a novel tool to identify somatic amino acid mutation clustering within proteins while taking into account protein structure.

License GPL-2

Depends R(>= 2.15), scatterplot3d, Biostrings, pwalign, multtest

Imports grDevices, graphics, stats, gdata

biocViews Clustering, Proteomics

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

git_branch devel

git_last_commit 900bc27

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-10-17

Contents

iPAC-package *Identification of Protein Amino acid mutation Clustering*

Description

This package finds mutation clusters on the amino acid level while taking into acount the protein structure.

Details

The *iPAC* package finds clusters of non-synonomous amino acid mutations via a two step process. First, *iPAC* links the amino acid positions in 3D space with their "canonical" ordering. The canonical ordering is considered to be the sequence of amino acids when the protein is viewed in linear form. Next, the canonical protein ordering is used to match the 3D positional information with the mutation file. The mutation file is a matrix of 0's (no mutation) and 1's (mutation) where each column represents an amino acid and each row represents an individual sample (test subject, cell line, etc). Thus if column i in row j had a 1, that would mean that the ith amino acid for person j had a nonsynonomous mutation. The last pre-processing step then maps the amino acids onto a 1D space while preserving (as best as possible) local neighbor relationships. The NMC algorithm (see *Ye. et. al.*) is then executed to identify the significant mutational clusters. Finally, the information is mapped back into the original 3D space and reported back to the user.

Author(s)

Author: Gregory A. Ryslik, Hongyu Zhao

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

References

Ye et. al., Statistical method on nonrandom clustering with application to somatic mutations in cancer. *BMC Bioinformatics*. 2010. doi:10.1186/1471-2105-11-11.

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

Borg, I., Groenen, P. (2005). Modern Multidimensional Scaling: theory and applications (2nd ed.). New York: Springer-Verlag.

Examples

```
## Not run:
#Extract the data from a CIF file and match it up with the canonical protein sequence.
#Here we use the 3GFT structure from the PDB, which corresponds to the KRAS protein.
CIF<-"https://files.rcsb.org/view/3GFT.cif"
Fasta<-"https://www.uniprot.org/uniprot/P01116-2.fasta"
KRAS.Positions<-get.AlignedPositions(CIF,Fasta, "A")

#Load the mutational data for KRAS. Here the mutational data was obtained from the
#COSMIC database (version 58).
data(KRAS.Mutations)

#Identify and report the clusters.
```

```
ClusterFind(mutation.data=KRAS.Mutations,
       position.data=KRAS.Positions$Positions,
       create.map = "Y",Show.Graph = "Y")

## End(Not run)
```

ClusterFind *ClusterFind*

Description

ClusterFind is the main method of the *iPAC* package. It identifies clusters of mutated amino acids while taking into account the protein structure.

Usage

```
ClusterFind(mutation.data, position.data, method = "MDS", alpha = 0.05,
   MultComp = "Bonferroni", Include.Culled = "Y", Include.Full = "Y",
   create.map = "Y", Show.Graph = "Y", Graph.Output.Path = NULL,
   Graph.File.Name = "Map.pdf", Graph.Title = "Mapping",
   OriginX = min(position.data[, 4]), OriginY = min(position.data[, 5]),
   OriginZ = min(position.data[, 6]))
```

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 column i in row j had a 1, that would mean that the ith amino acid for person j had a nonsynonomous mutation.
position.data	A dataframe consisting of five 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* for further information on how to construct this matrix.
method	You can select whether you want a "MDS" or "Linear" approach in order to map the protein into a 1D space.
alpha	The significance level used in the NMC calculation. Please see *Ye. et. al.* for more information.
MultComp	The multiple comparisons adjustment used in the NMC calculation. Possible options are "None", "Bonferroni" and "BH". Please see *Ye. et. al.* for more information.
Include.Culled	If "Y", the standard NMC algorithm will be run on the protein after removing the amino acids for which there is no positional data.
Include.Full	If "Y", the standard NMC algorithm will be run on the full protein sequence.
create.map	If "Y", a graphical representation of the the dimension reduction from 3D to 1D space will be created (though not necessarily displayed).
Show.Graph	If "Y", the graph representation will be displayed. Warning: You must be running R in a GUI environment, otherwise, an error will occur.

`Graph.Output.Path`
: If you would like the picture saved atomatically to the disk, specify the output directory here. The *Graph.File.Name* variable must be set as well.

`Graph.File.Name`
: If you would like the picture saved automatically to the disk, specify the output file name. The *Graph.Output.Path* variable must be set as well.

`Graph.Title`
: The title of the graph to be created.

`OriginX`
: If the "Linear" method is chosen, this specifies the x-coordinate part of the fixed point.

`OriginY`
: If the "Linear" method is chosen, this specifies the y-coordinate part of the fixed point.

`OriginZ`
: If the "Linear" method is chosen, this specifies the z-coordinate part of the fixed point.

Details

The linear method fixes a point, defined by the parameters *OriginX*, *OriginY*, *OriginZ*, and then calculates the distance from each amino acid to that point. The graph produced by *ClusterFind* (if requested), shows these distances as dotted green lines. The length of the green line is used to reorder the protein, with the amino acid that corresponds to the shortest green line being ordered first and the amino acid corresponding to the longest green line being ordered last.

Additional methods will be available in future versions of this package.

Value

`Remapped`
: This shows the clusters found while taking the 3D structure into account.

`OriginalCulled`
: This shows the clusters found if you run the NMC algorithm on the canonical linear protein, but with the amino acids for which we don't have 3D positional data removed.

`Original`
: This shows the clusters found if you run the NMC algorithn on the canonical linear protein with all the amino acids.

`MissingPositions`
: This shows which amino acids are present in the mutation matrix but for which we do not have positions. These amino acids are cut from the protein when calculating the *Remapped* and *OriginalCulled* results.

Note

If no significant clusters are found, a "NULL" will be returned for the appropriate section (Remapped, OriginalCulled, or Original).

If you want the graph to just display on a new R graphics devices without saving it to the disk, simply set the *Graph.Output.Path* or the *Graph.File.Name* parameters to be NULL while leaving both the *create.map* and the *Show.Graph* parameters to be "Y".

If you are running this algorithm on a terminal with no GUI (such as a computational cluster), set *Show.Graph* to "N" as R will not be able to open a new graphics device. However, you can still have the graphics saved for later viewing by setting the *Graph.Output.Path* and *Graph.File.Name* variables.

If *ClusterFind* displays the message "Error in 0:(n - j) : NA/NaN argument", this most likely signifies that after removing the amino acids for which there is no positional data, the mutation data matrix is all 0's. For instance, if all the mutations occured on the 5th amino acid in the protein, and

we did not have 3D positional information for that amino acid, the mutation data for the remaining positions would be all 0's. An error message in this situation will be displayed in the results section as well. In such cases, the user should run the original NMC algorithm only (see *nmc*) or select an alternative protein structure.

When unmapping back to the original space, the end points of the cluster in the mapped space are used as the endpoints of the cluster in the unmapped space.

References

Ye et. al., Statistical method on nonrandom clustering with application to somatic mutations in cancer. *BMC Bioinformatics*. 2010. doi:10.1186/1471-2105-11-11.

Examples

```
#Extract the data from a CIF file and match it up with the canonical protein sequence.
#Here we use the 3GFT structure from the PDB, which corresponds to the KRAS protein.
CIF<-"https://files.rcsb.org/view/3GFT.cif"
Fasta<-"https://www.uniprot.org/uniprot/P01116-2.fasta"
KRAS.Positions<-get.Positions(CIF,Fasta, "A")

#Load the mutational data for KRAS. Here the mutational data was obtained from the
#COSMIC database (version 58).
data(KRAS.Mutations)

#Identify and report the clusters using the default MDS method.
ClusterFind(mutation.data=KRAS.Mutations,
       position.data=KRAS.Positions$Positions,
       create.map = "Y",Show.Graph = "Y")

#Identify and report the clusters using the linear method.
ClusterFind(mutation.data=KRAS.Mutations,
       position.data=KRAS.Positions$Positions,
       create.map = "Y",Show.Graph = "Y", method = "Linear")
```

`get.AASeq` *get.AASeq*

Description

Reads a FASTA (or similarly formatted) file and returns all the amino acids as a list.

Usage

```
get.AASeq(filelocation)
```

Arguments

`filelocation`	The location of the file being read.

Details

The first line in the file being processed must begin with a ">". If the ending character in the file is a "*" (indicating a terminator – as is common in the COSMIC database), this character is excluded from the sequence as well.

Value

An ordered list of amino acids in the protein. Each element of the list represents an amino acid.

Examples

```
#Reads the amino acid sequence for KRAS from Uniprot
get.AASeq("https://www.uniprot.org/uniprot/P01116-2.fasta")
```

`get.AlignedPositions` *get.AlignedPositions*

Description

This function reads a CIF file to extract the names and (x,y,z) coordinates of each residue. It then performs a pairwise alignment to convert the amino acid ordering in the CIF file to the canonical ordering specified by the FASTA file. The first element in the returned list, $Positions, is the positions matrix required by the ClusterFind method.

Usage

```
get.AlignedPositions(CIF.File.Location, Fasta.File.Location, chain.required = "A",
       RequiredModelNum = NULL, patternQuality = PhredQuality(22L),
       subjectQuality = PhredQuality(22L), type = "global-local",
       substitutionMatrix = NULL, fuzzyMatrix = NULL, gapOpening = -10,
       gapExtension = -4, scoreOnly = FALSE)
```

Arguments

`CIF.File.Location`
: The location of the CIF file to be read.

`Fasta.File.Location`
: The location of the FASTA (or FASTA-like) file to be read.

`chain.required`
: The side chain in the protein from which to extract positions in the CIF file.

`RequiredModelNum`
: The required model num to extract positions from in the CIF file. If the RequiredModelNum == NULL, the method will use the first model number found in the file.

`patternQuality`
: The patternQuality parameter in the pairwiseAlignment function in the pwalign package.

`subjectQuality`
: The subjectQuality parameter in the pairwiseAlignment function in the pwalign package.

`type`
: The type parameter in the pairwiseAlignment function in the pwalign package. This should NOT be changed from "global-local" as we use the canonical protein from the FASTA file as the global pattern and the extracted positions from the CIF as the subject pattern. We then attempt to align parts of the subject pattern to the entire global pattern.

`substitutionMatrix`
: The substitutionMatrix parameter in the pairwiseAlignment function in the pwalign package.

`fuzzyMatrix`	The fuzzyMatrix parameter in the pairwiseAlignment function in the pwalign package.
`gapOpening`	The gapOpening parameter in the pairwiseAlignment function in the pwalign package.
`gapExtension`	The gapExtension parameter in the pairwiseAlignment function in the pwalign package.
`scoreOnly`	The scoreOnly parameter in the pairwiseAlignment function in the pwalign package.

Details

This method is currently in BETA and is provided only as a convenient way to extract the required 3D positional information from a CIF file. Currently, CIF (and PDB) files can have a number of deviations from the canonical protein sequence including additional, missing and mismatched amino acids. The amino acid numbering in CIF files can also be different from the canonical protein. This makes it difficult to match up mutational data (from sources such as COSMIC) to 3D positional data (from sources such as the PDB) and necessitates the use of this function.

This method extracts the canonical amino acid sequence from the file at *Fasta.File.Location*. It then attempts to align the amino acids extracted from the CIF file to the canonical sequence using the pairwiseAlignment function in the package *pwalign* that is available on *Bioconductor*. After alignment, any amino acids that are mismatched between the canonical sequence and the extracted sequence are automatically removed so that the *ClusterFind* method, which requires positional data as input, is only run on those amino acids which are correctly matched.

Value

`Positions`	A dataframe that shows the extracted amino acids, their numerical position in the protein order, the protein side chain being used and the amino acid positions in 3D space.
`Diff.Count`	A check that the amino acids remaining are in fact matched to the canonical protein. This returns a count of the number of amino acids remaining that do not match the canonical sequence and should be 0 if a successful alignment occurred.
`Diff.Positions`	A description of the mismatched amino acids if any were found. If a succesful alignment occurred, this will be NULL.
`Alignment.Result`	The raw alignment result returned by the pairwiseAlignment method in the *pwalign* package.
`Result`	The final status of the alignment. If it is "OK", that means that this alignment appears to be ok. If the alignment failed, this item will contain an error message.

Note

This function is provided as a convenience to the user so that they may easily obtain the required input for the *ClusterFind* function. The user should validate the results.

The "Diff.Count" and "Diff.Positions" parts of the returned value should always be "0" and "NULL" if the positions were successfully extracted as mismatched positions are dropped automatically. This behavior is different from the *get.Positions* function where a reconciliation of differences is attempted using the information within the CIF file.

If you would like to contribute to this function, please contact the author.

References

Biostrings Package. 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

Examples

```
#Observe that position 61 is missing. It is atuomatically dropped as the pdb data
#specifies it as a "H" while the FASTA sequence specifies it as "Q".
CIF<-"https://files.rcsb.org/view/3GFT.cif"
Fasta<-"https://www.uniprot.org/uniprot/P01116-2.fasta"
get.AlignedPositions(CIF,Fasta, "A")
```

`get.Positions` *get.Positions*

Description

This function reads a CIF file to extract the names and (x,y,z) coordinates of each residue. Then, the information within the CIF file itself is used to identify the canonical position for each amino acid.

Usage

```
get.Positions(CIF.File.Location, Fasta.File.Location, chain.required = "A",
     RequiredModelNum = NULL)
```

Arguments

`CIF.File.Location`
: The location of the CIF file to be read.

`Fasta.File.Location`
: The location of the FASTA (or FASTA-like) file to be read.

`chain.required`
: The side chain in the protein from which to extract positions in the CIF file.

`RequiredModelNum`
: The required model num to extract positions from in the CIF file. If the RequiredModelNum == NULL, the method will use the first model number found in the file.

Details

This algorithm attempts to recreate the canonical sequence by considering 3 seperate parts of a CIF file. The first part, the "_pdbx_poly_seq_scheme" section corresponds to the SEQRES entries in PDB files and lists all the amino acid residues in the protein that were studied including those that were not included in the 3D structure. The second part, the "struct_ref_seq" section, links the Uniprot (which provides the canonical sequence), PDB and internal CIF amino acid numbering schemes. Finally, the "_atom_site" section provides the (x,y,z) positional information. All three parts are tied together via a "seq_id" numerical identifier. It is thus possible to use the numbering scheme map provided in the "struct_ref_seq" section to calculate the canonical numbering for the residues listed in the "_pdbx_poly_seq_scheme" and then finally match them up to the residues listed in the "_atom_site" section.

Value

Positions	A dataframe that shows the amino acids extracted, their position in protein ordering, the side chain and their position in 3D space.
External.Mismatch	This shows the mismatches that were found between the amino acids extracted from the file and the canonical sequence in the FASTA file.
PDB.Mismatch	This shows the mismatches that are known to exist using only the information available in the CIF file.
Result	The final status of extracting the amino acid sequence. It will either display "OK" or an error message.

Note

Due to the complexity of CIF file, this function is only able to account for a subset of the structures in the PDB database. If this function fails, consider using the the *get.AlignedPositions* function.

If you would like to contribute to this function, please contact the author.

Examples

```
##################################################
#Observe that the final result from the code below is "OK". That is because the only
#mismatched residue at position 61, was documented in the CIF file as well.
#Thus it is considered a "reconciled" mismatch. It is up to the user to decide if
#they want to include it in the position sequence or remove it.

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

##################################################
#Observe that the final result from the code below is "FAILURE". For PIK3CA there were
#10 mismatched residues between the CIF file and the canonical sequence.
#However, none of these residues are explained within the actual CIF file.

CIF<- "https://files.rcsb.org/view/2RD0.cif"
Fasta<-"https://cancer.sanger.ac.uk/cosmic/sequence?width=700&ln=PIK3CA&type=protein&height=500"
PIK3CA.extracted.positions<- get.Positions(CIF,Fasta , "A")
##################################################


##################################################
#Observe that the final result from the code below is "OK". Here we use a different file
#location for the canonical sequence -- the UNIPROT database. The canonical sequence is slightly
#different and matches up exactly to the extracted positions. As there is only 1 isoform listed
#on UNIPROT for PIK3CA we suggest that users obtain the mutational data and the canonical
#sequence information from the same source. For example, if your mutation data was obtained from
#COSMIC, you should use COSMIC to get the canonical protein sequence.

CIF <- "https://files.rcsb.org/view/2RD0.cif"
Fasta <- "https://www.uniprot.org/uniprot/P42336.fasta"
PIK3CA.extracted.positions<- get.Positions(CIF, Fasta, "A")
##################################################
```

get.Remapped.Order *get.Remapped.Order*

Description

get.Remapped.Order returns the reordering of the culled mutation matrix using the remapper of choice (linear vs MDS).

Usage

```
get.Remapped.Order(mutation.data, position.data, method = "MDS", OriginX = min(position.data[,4]),
```

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 column i in row j had a 1, that would mean that the ith amino acid for person j had a nonsynonomous mutation.
position.data	A dataframe consisting of five 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* for further information on how to construct this matrix.
method	You can select whether you want a "MDS" or "Linear" approach in order to map the protein into a 1D space.
OriginX	If the "Linear" method is chosen, this specifies the x-coordinate part of the fixed point.
OriginY	If the "Linear" method is chosen, this specifies the y-coordinate part of the fixed point.
OriginZ	If the "Linear" method is chosen, this specifies the z-coordinate part of the fixed point.

Details

The culled mutation matrix is the mutation matrix after the amino acids that have missing positional data have been removed. The amino acid positions that have missing positional data will not be displayed in the result.

Note

The returned value shows the remapped order of the culled mutation matrix.

This method is still in beta. If any bugs found, please email the package authors.

Examples

```
#Extract the data from a CIF file and match it up with the canonical protein sequence.
#Here we use the 3GFT structure from the PDB, which corresponds to the KRAS protein.
CIF<-"https://files.rcsb.org/view/3GFT.cif"
Fasta<-"https://www.uniprot.org/uniprot/P01116-2.fasta"
KRAS.Positions<-get.AlignedPositions(CIF,Fasta, "A")
```

```
#Load the mutational data for KRAS. Here the mutational data was obtained from the
#COSMIC database (version 58).
data(KRAS.Mutations)

#Show the remapped order using the MDS remapper.
get.Remapped.Order(KRAS.Mutations, KRAS.Positions$Positions)

#Show the remapped order using the Linear remapper. Note that Amino Acid 61 is missing
#if get.AlignedPositions was used. Thus Amino Acid 61 is missing in the returned result.
get.Remapped.Order(KRAS.Mutations, KRAS.Positions$Positions, method = "Linear")
```

`get.SingleLetterCode` *get.SingleLetterCode*

Description

This function takes in the 3 letter amino acid representation and returns the 1 letter representation.

Usage

```
get.SingleLetterCode(code)
```

Arguments

code	The 3 letter code in all capital letters.

Details

If the 3 letter code is not found (either because it does not exist or because it is not in all capital letters), an "X" is returned.

This function is used as a helper function of *get.AlignedPositions* and *get.Positions*. It is not necessary to invoke it independently and we include it here solely as a matter of convenience for the user.

Value

An amino acid single letter code returned as a string. The string will be an element in:

{"P","A","V","L","I","M","C","F","Y","W","H","K","R","Q","N","E","D","S","T","X"}.

Examples

```
get.SingleLetterCode("MET")

get.SingleLetterCode("GLN")

get.SingleLetterCode("gln")

get.SingleLetterCode("DMY")
```

KRAS.Mutations *KRAS.Mutations*

Description

A matrix of the verified somatic mutations for KRAS from the COSMIC database.

Usage

```
data(KRAS.Mutations)
```

Format

There are 149 rows, each representing a different sample and 188 colums – one for each amino acid in the protein.

Source

Version 58 of the cosmic database was used.

References

S A Forbes, G Bhamra, S Bamford, E Dawson, C Kok, J Clements, A Menzies, J W Teague, P A Futreal, and M R Stratton. The catalogue of somatic mutations in cancer (COSMIC). Current Protocols in Human Genetics / Editorial Board, Jonathan L. Haines ... [et Al.], Chapter 10:Unit 10.11, April 2008. ISSN 1934-8258. doi: 10.1002/0471142905.hg1011s57.

URL http://www.ncbi.nlm.nih.gov/pubmed/18428421. PMID: 18428421.

Examples

```
data(KRAS.Mutations)

#Show the first 10 samples
KRAS.Mutations[1:10,]

#Show which samples had position 12 mutated
KRAS.Mutations[,12]
```

nmc *nmc*

Description

This runs the clustering algorithm developed by Ye. et. al on the original linear protein.

Usage

```
nmc(x, alpha = 0.05, multtest = c("Bonferroni", "BH", "None"))
```

Arguments

x	x is a matrix of 0's and 1's. Each column represents a sample (cell-line, individual, etc.) while each column represents an amino acid in the protein. The i-th column represents the i-th amino acid.
alpha	The significance level to be used.
multtest	The type of multiple comparisons adjustment to be used.

Value

Returns a matrix of results which include the starting and ending positions of the clusters along with a p-value for each cluster.

Note

Please see the original paper and methodology for a more detailed description of each of the parameters to this function.

References

Ye et. al., Statistical method on nonrandom clustering with application to somatic mutations in cancer. *BMC Bioinformatics*. 2010. doi:10.1186/1471-2105-11-11.

Examples

```
data(KRAS.Mutations)
nmc(KRAS.Mutations, alpha = 0.05, multtest = "Bonferroni")
```

PIK3CA.Mutations *PIK3CA.Mutations*

Description

A matrix of the verified somatic mutations for PIK3CA from COSMIC.

Usage

```
data(PIK3CA.Mutations)
```

Format

There are 96 rows, each representing a sample mutation and 1068 colums – one for each amino acid in the protein.

Source

Version 58 of the cosmic database was used.

References

S A Forbes, G Bhamra, S Bamford, E Dawson, C Kok, J Clements, A Menzies, J W Teague, P A Futreal, and M R Stratton. The catalogue of somatic mutations in cancer (COSMIC). Current Protocols in Human Genetics / Editorial Board, Jonathan L. Haines ... [et Al.], Chapter 10:Unit 10.11, April 2008. ISSN 1934-8258. doi: 10.1002/0471142905.hg1011s57.

URL http://www.ncbi.nlm.nih.gov/pubmed/18428421. PMID: 18428421.

Examples

```
data(PIK3CA.Mutations)

#Show the first 10 samples
PIK3CA.Mutations[1:10,]

#Show which samples had position 1047 mutated
PIK3CA.Mutations[,1047]
```

Plot.Protein.Linear *Plot.Protein.Linear*

Description

Plot.Protein.Linear creates a visual representation of the protein reordering by showing the protein in linear form and setting the color based upon the remapped amino acid position.

Usage

```
Plot.Protein.Linear(path, colCount, cex=0.5, height = 1,width = 1,
          title = "Protein Reordering", color.palette = "heat")
```

Arguments

path	The remapped protein using only those amino acids for which mutational data is available.
colCount	How many columns you want in the plot
cex	How large you want the text to be. See R graphical parameters for more information.
height	The height of the resulting box that outlines the amino acid number.
width	The width of the resulting box that outlines the amino acid number.
title	The title that you want the graph to display.
color.palette	One of of the standard color palettes available in R. The options are "heat", "gray", "topo", and "cm".

Details

The terminal you are running this on must be able to create an R graphical device.

Note

This method is still in beta. If any bugs found, please email the package authors.

Examples

```
#Extract the data from a CIF file and match it up with the canonical protein sequence.
#Here we use the 3GFT structure from the PDB, which corresponds to the KRAS protein.
CIF<-"https://files.rcsb.org/view/3GFT.cif"
Fasta<-"https://www.uniprot.org/uniprot/P01116-2.fasta"
KRAS.Positions<-get.Positions(CIF,Fasta, "A")

#Load the mutational data for KRAS. Here the mutational data was obtained from the
#COSMIC database (version 58).
data(KRAS.Mutations)

#Show the remapped order using the MDS remapper.
new.ordering<-get.Remapped.Order(KRAS.Mutations, KRAS.Positions$Positions)

#Create the Plots. Note that Amino Acid 61 is missing if get.AlignedPositions
#was used. Thus Amino Acid 61 is missing in the plots below.
Plot.Protein.Linear(new.ordering, 25, color.palette = "heat")
Plot.Protein.Linear(new.ordering, 25, color.palette = "gray")
```

Index