Package ‘Autoplotprotein’

January 20, 2025

Type Package

Title Development of Visualization Tools for Protein Sequence

Version 1.1

Date 2017-06-02

Author Xiaoyu Zhang

Maintainer Yao Geng `<gengyao0103521@qq.com>`

Description The image of the amino acid transform on the protein level is drawn, and the automatic routing of the functional elements such as the domain and the mutation site is completed.

License GPL-3

Depends XML, plyr, plotrix, seqinr, ade4

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-06 10:00:07 UTC

Contents

`Autoplotprotein-package` *Development of Visualization Tools for Protein Sequence*

Description

The image of the amino acid transform on the protein level is drawn, and the automatic routing of the functional elements such as the domain and the mutation site is completed.

Details

The DESCRIPTION file:

Package:	Autoplotprotein
Type:	Package
Title:	Development of Visualization Tools for Protein Sequence
Version:	1.1
Date:	2017-06-02
Author:	Xiaoyu Zhang
Maintainer:	Yao Geng <gengyao0103521@qq.com>
Description:	The image of the amino acid transform on the protein level is drawn, and the automatic routing of the functiona
License:	GPL-3
Depends:	XML, plyr, plotrix, seqinr, ade4

Index of help topics:

```
Autoplotprotein         Two - dimensional structure of protein
Autoplotprotein-package
                        Development of Visualization Tools for Protein
                        Sequence
conservation            conservation
data                    Save the information
domain_data             downloading protein length
length_data             downloading protein length
plotdomain              ploting domain
plotmutagensis          ploting mutagensis
plotsite                ploting site
site_data               downloading protein site
```

Author(s)

Xiaoyu Zhang

Maintainer: Yao Geng <gengyao0103521@qq.com>

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

codehelp

`Autoplotprotein` *Two - dimensional structure of protein*

Description

Draw a visualized structure of the protein

Usage

```
Autoplotprotein()
```

Details

The tool ennable visualization of amino acid changes at the protein level,The scale of a protein domain and the position of a functional motif/site will be precisely defined

Value

Visualization of protein structure

Author(s)

Xiaoyu Zhang

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

codehelp

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function ()
{
    library("ade4")
    library("seqinr")
    library("plotrix")
    protein = read.table("Protein.txt", sep = "\t", stringsAsFactors = F)
    domain = read.table("Domain.txt", sep = "\t", stringsAsFactors = F)
```

```
length = read.table("Length.txt", sep = "\t", stringsAsFactors = F)
site = read.table("Site.txt", sep = "\t", stringsAsFactors = F)
muta = read.table("Mutagenesis.txt", sep = "\t", stringsAsFactors = F)
option = read.table("Option.txt", sep = "\t", stringsAsFactors = F)
zoomin = read.table("ZoomIn.txt", sep = "\t", stringsAsFactors = F)
size <- c(10.5, 7.27)
high <- c(1, -1)
sizen = size[1]
highn = high[1]
if (option[2, 2] == "no") {
    sizen = size[2]
    highn = high[2]
}
path = protein[1]
pdf(as.character(path), height = sizen[1], width = 11)
layout(matrix(c(1, 2), nrow = 1), widths = c(1, 3))
par(oma = c(3, 0, 2, 0), mar = c(4, 0, 2, 0) + 0.4)
nameOfYourQuery = option[2, 1]
additionalOptions = option[2, 2]
showReferenceSequence = option[2, 3]
showConservationScore = option[2, 4]
showGridlinesAtTicks = option[2, 5]
conservation = option[2, 6]
zoomIn = zoomin[2, 1]
zoomStart = zoomin[2, 2]
zoomEnd = zoomin[2, 3]
tickSize = as.numeric(zoomin[2, 4])
plot((-30:-15), rep(-1, 16), col = "white", type = "l", ann = FALSE,
    bty = "n", xaxt = "n", yaxt = "n", xlim = c(-160, -15),
    ylim = c(highn[1], -5.5))
if (additionalOptions == "yes") {
    if (conservation == "yes") {
        lines((-30:-15), rep(0, 16), col = "purple3")
        lines((-30:-15), rep(-0.5, 16), col = "purple3")
        lines((-30:-15), rep(-1, 16), col = "purple3")
        text(-100, -0.5, "Conservation", col = "purple3",
            cex = 0.9, font = 2)
        text(-45, -1, "1", col = "purple3", cex = 0.9)
        text(-45, -0.5, "0.5", col = "purple3", cex = 0.9)
        text(-45, 0, "0", col = "purple3", cex = 0.9)
    }
}
if (additionalOptions == "yes") {
    if (showReferenceSequence == "yes") {
        text(-100, -4.9, "Reference", col = "black", cex = 0.9,
            font = 2)
    }
}
if (additionalOptions == "yes") {
    if (showConservationScore == "yes") {
        text(-100, 0.5, "Score", col = "purple3", cex = 0.9,
            font = 2)
    }
```

```
}
text(-100, -2.95, nameOfYourQuery, col = "blue", cex = 0.9,
    font = 2)
Protein = function(start = 1, end, height = -0.3, color = "green",
    face = "stereoscopic") {
    x = 0
    kong1 = (round(log(start, 10)) + 1) * start/50
    kong2 = (round(log(end, 10)) + 1) * end/50
    if (round(log(end, 10)) + 1 <= 5) {
        kong2 = (round(log(end, 10)) + 1) * end/50
    }
    else {
        kong2 = 5 * end/50
    }
    h1 = -2.8
    h2 = -3.1
    boxplot((1:as.numeric(end)), rep(h1, as.numeric(end)),
        xlab = "Amino Acid Position", ylab = "", xlim = c(0,
            as.numeric(end)), ylim = c(highn[1], -5.5), axes = FALSE)
    if (face == "stereoscopic") {
        cylindrect(start, h1, end, h2, col = color, gradient = "y")
    }
    else {
        rect(start, h1, end, h2, col = color)
    }
    text(0, h1 - height/2, start, adj = 1)
    text(end - 17, h1 - height/2, end, adj = 0)
}
ZoomIn = function(start = 1, end, height = -0.3, color = "green",
    face = "stereoscopic", zoomstart, zoomend) {
    x = 0
    kong1 = (round(log(start, 10)) + 1) * start/50
    kong2 = (round(log(end, 10)) + 1) * end/50
    if (round(log(end, 10)) + 1 <= 5) {
        kong2 = (round(log(end, 10)) + 1) * end/50
    }
    else {
        kong2 = 5 * end/50
    }
    h1 = -2.8
    h2 = -3.1
    boxplot((as.numeric(zoomstart):as.numeric(zoomend)),
        rep(h1, as.numeric(zoomend)), xlab = "Amino Acid Position",
        ylab = "", xlim = c(as.numeric(zoomstart), as.numeric(zoomend)),
        ylim = c(highn[1], -5.5), axes = FALSE)
    if (face == "stereoscopic") {
        cylindrect(start, h1, end, h2, col = color, gradient = "y")
    }
    else {
        rect(start, h1, end, h2, col = color)
    }
    text(start, h1 + height/2, start, adj = 1)
    text(end, h1 + height/2, end, adj = 0)
    }
    if (zoomIn == "yes") {
        ZoomIn(start = as.numeric(length[1]), end = as.numeric(length[2]),
            height = as.numeric(protein[4]), color = as.character(protein[5]),
            face = protein[6], zoomstart = zoomin[2, 2], zoomend = zoomin[2,
                3])
    }
    else {
        Protein(start = as.numeric(length[1]), end = as.numeric(length[2]),
            height = as.numeric(protein[4]), color = as.character(protein[5]),
            face = protein[6])
    }
    legend("topleft", legend = c("mutation", "Protein Domain"),
        pch = c(19, 15), col = c("lightseagreen", "deeppink"),
        box.col = "white", bg = "white", pt.cex = 1.5, text.width = 1)
    ticks = seq(0, as.numeric(length[2]), by = tickSize)
    axis(side = 1, at = ticks, las = 3)
    if (additionalOptions == "yes") {
        if (showGridlinesAtTicks == "yes") {
            len = array(rep(1:as.numeric(length[2])))
            for (i in 1:length(len)) {
                abline(v = ticks[i], lty = 3, lwd = 0.5, col = "lightgray")
            }
        }
    }
}
```

`conservation` *conservation*

Description

Draw a conservative curve, calculate the conservative score

Usage

```
conservation()
```

Details

The tool ennable visualization of amino acid changes at the protein level,The scale of a protein domain and the position of a functional motif/site will be precisely defined. The features available includeting conservation, conservation score

Value

The returned value is a conservative score

Author(s)

Xiaoyu Zhang

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

help

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function ()
{
    protein = read.table("Protein.txt", sep = "\t", stringsAsFactors = F)
    domain = read.table("Domain.txt", sep = "\t", stringsAsFactors = F)
    length = read.table("Length.txt", sep = "\t", stringsAsFactors = F)
    site = read.table("Site.txt", sep = "\t", stringsAsFactors = F)
    muta = read.table("Mutagenesis.txt", sep = "\t", stringsAsFactors = F)
    option = read.table("Option.txt", sep = "\t", stringsAsFactors = F)
    zoomin = read.table("ZoomIn.txt", sep = "\t", stringsAsFactors = F)
    nameOfYourQuery = option[2, 1]
    additionalOptions = option[2, 2]
    showReferenceSequence = option[2, 3]
    showConservationScore = option[2, 4]
    showGridlinesAtTicks = option[2, 5]
    conservation = option[2, 6]
    zoomIn = zoomin[2, 1]
    zoomStart = zoomin[2, 2]
    zoomEnd = zoomin[2, 3]
    tickSize = as.numeric(zoomin[2, 4])
    referenceSequencePositionInFile = option[2, 7]
    option = read.table("Option.txt", sep = "\t", stringsAsFactors = F)
    a <- read.fasta(file = "alignmentFile.fasta")
    seq <- list()
    for (i in 1:length(a)) {
        seq[[i]] <- a[[i]][1:length(a[[i]])]
    }
    numberOfSeq <- length(seq)
    mat <- matrix(0, nrow = length(a), ncol = length(a[[1]]))
    for (i in 1:length(seq)) {
        mat[i, ] <- seq[[i]]
    }
    df <- as.data.frame(mat)
    tdf <- t(df)
    referenceSequencePositionInFile = option[2, 7]
    referenceSeq <- tdf[which(tdf[, as.numeric(referenceSequencePositionInFile)] !=
        "-"), ]
    referenceSeq <- as.data.frame(referenceSeq)
    write.table(referenceSeq, file = "alignment_table", sep = "\t",
        quote = F, row.names = F, col.names = F)
    counter <- rep(0, nrow(referenceSeq))
    a <- read.table("alignment_table", sep = "\t")
    a <- data.frame(lapply(a, as.character), stringsAsFactors = FALSE)
    for (i in 1:nrow(a)) {
        a[i, "consensus"] <- paste(as.character(a[i, ]), collapse = "")
    }
    countBases <- function(string) {
        table(strsplit(string, "")[[1]])
    }
    c <- as.character(a[, "consensus"])
    tab <- list()
    for (i in 1:length(c)) {
        tab[[i]] <- countBases(c[i])
    }
    score <- rep(0, nrow(a))
    for (i in 1:length(tab)) {
        for (j in 1:length(tab[[i]])) {
         if ((names(tab[[i]][j])) == a[i, ][as.numeric(referenceSequencePositionInFile)])
                score[i] <- tab[[i]][j]
        }
    }
    scorePlot <- -(((score/numberOfSeq)))
    a <- read.fasta(file = "alignmentFile.fasta")
    seqForPlot <- a[[as.numeric(referenceSequencePositionInFile)]][
which(a[[as.numeric(referenceSequencePositionInFile)]] !=
        "-")]
    if (additionalOptions == "yes") {
        if (conservation == "yes") {
            lines(scorePlot, col = "purple3")
        }
    }
    if (additionalOptions == "yes") {
        if (showReferenceSequence == "yes") {
            rect(0, -4.75, length(scorePlot), -5.05, col = "white",
                border = NA)
            for (i in 1:length(seqForPlot)) {
                text(i, -4.9, toupper(seqForPlot[i]), font = 2,
                  cex = 1)
            }
        }
    }
    if (additionalOptions == "yes") {
        if (showConservationScore == "yes") {
            rect(0, 0.3, length(scorePlot), 0.7, col = "white",
                border = NA)
            for (i in 1:length(seqForPlot)) {
                text(i, 0.5, toupper(abs(round(scorePlot[i],
                  1))), font = 2, cex = 0.8, srt = 90, col = "purple3")
            }
        }
    }
  }
```

`data` *Save the information*

Description

Keep all the information of the painted protein in a file

Usage

```
data()
```

Details

Save information, including protein mutation point information, domain information, option information, enlargement information, protein information, length information and site information

Value

Data of various kinds of information

Author(s)

Xiaoyu Zhang

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

codehelp

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function ()
{
    library("ade4")
    library("seqinr")
    library("plotrix")
    protein = read.table("Protein.txt", sep = "\t", stringsAsFactors = F)
    domain = read.table("Domain.txt", sep = "\t", stringsAsFactors = F)
    length = read.table("Length.txt", sep = "\t", stringsAsFactors = F)
    site = read.table("Site.txt", sep = "\t", stringsAsFactors = F)
    muta = read.table("Mutagenesis.txt", sep = "\t", stringsAsFactors = F)
    option = read.table("Option.txt", sep = "\t", stringsAsFactors = F)
    zoomin = read.table("ZoomIn.txt", sep = "\t", stringsAsFactors = F)
    c <- merge(muta, domain, all = T, sort = FALSE)
    c <- merge(c, option, all = T, sort = FALSE)
    c <- merge(c, zoomin, all = T, sort = FALSE)
    c <- merge(c, protein, all = T, sort = FALSE)
    c <- merge(c, length, all = T, sort = FALSE)
    c <- merge(c, site, all = T, sort = FALSE)
    write.table(c, file = "data.txt", sep = "\t", quote = FALSE,
        row.names = F, col.names = F)
  }
```

domain_data *downloading protein length*

Description

Load the start and end positions of the domain

Usage

```
domain_data()
```

Details

The tool ennable visualization of amino acid changes at the protein level,The scale of a protein domain and the position of a functional motif/site will be precisely defined. The features available include domains

Value

The start and end positions of the domain

Author(s)

Xiaoyu Zhang

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

codehelp

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function ()
{
    library(XML)
    library(plyr)
    protein = read.table("Protein.txt", sep = "\t", stringsAsFactors = F)
    name = protein[2]
    url_p = "http://www.uniprot.org/uniprot/"
    url_s = "#showFeatures"
    url_w = paste(url_p, name, url_s, sep = "")
    url = url_w
    doc <- htmlParse(url)
    position_d = xpathSApply (doc, "//table[@id= 'domainsAnno_section']
  /tr/td/  a[@class = 'position tooltipped']",
        xmlValue)
   name_d = xpathSApply (doc, "//table[@id= 'domainsAnno_section']/tr/td/span[@property='text']",
        xmlValue)
    s_d = c()
    for (i in 1:length(position_d)) {
        s_d[i] <- gsub(pattern = "//D", replacement = "x", position_d[i])
    }
    s_d <- strsplit(s_d, "xxx")
    d1_d <- laply(s_d, function(x) x[1])
    d2_d <- laply(s_d, function(x) x[2])
    r1_d = d1_d
    r2_d = d2_d
    r3_d = name_d
    dfrm_d = data.frame(r1_d, r2_d, r3_d)
    write.table(dfrm_d, file = "Domain.txt", sep = "/t", quote = FALSE,
        row.names = F, col.names = F)
  }
```

`length_data` *downloading protein length*

Description

Download the length of the protein, including the starting and ending positions

Usage

```
length_data()
```

Details

Download the length of the protein, including the starting and ending positions

Value

The length of the protein

Author(s)

Xiaoyu Zhang

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

codehelp

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function ()
{
    library(XML)
    library(plyr)
    protein = read.table("Protein.txt", sep = "\t", stringsAsFactors = F)
    name = protein[2]
    url_p = "http://www.uniprot.org/uniprot/"
    url_s = "#showFeatures"
    url_w = paste(url_p, name, url_s, sep = "")
    url = url_w
    doc <- htmlParse(url)
    position_l = xpathSApply (doc, "//table[@id= 'peptides_section']
  /tr/td/  a[@class = 'position tooltipped']",
        xmlValue)
    s_l <- c()
    for (i in 1:length(position_l)) {
        s_l[i] <- gsub(pattern = "//D", replacement = "x", position_l[i])
    }
    s_l <- strsplit(s_l, "xxx")
    d2_l <- laply(s_l, function(x) x[2])
    r1_l <- 0
    r2_l <- d2_l
    dfrm_l <- data.frame(r1_l, r2_l)
    write.table(dfrm_l, file = "Length.txt", sep = "/t", quote = FALSE,
        row.names = F, col.names = F)
  }
```

---

plotdomain *ploting domain*

Description

Draw the domain of the protein

Usage

```
plotdomain()
```

Details

The tool ennable visualization of amino acid changes at the protein level,The scale of a protein domain and the position of a functional motif/site will be precisely defined. The features available include domains

Value

The starting position, end position and name of the protein domain

Author(s)

Xiaoyu Zhang

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

codehelp

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function ()
{
    protein = read.table("Protein.txt", sep = "\t", stringsAsFactors = F)
    domain = read.table("Domain.txt", sep = "\t", stringsAsFactors = F)
    length = read.table("Length.txt", sep = "\t", stringsAsFactors = F)
    site = read.table("Site.txt", sep = "\t", stringsAsFactors = F)
    muta = read.table("Mutagenesis.txt", sep = "\t", stringsAsFactors = F)
    option = read.table("Option.txt", sep = "\t", stringsAsFactors = F)
    zoomin = read.table("ZoomIn.txt", sep = "\t", stringsAsFactors = F)
```

```
Domain = function(start, end, name, height = -0.3, color = "orange",
    face = "stereoscopic", protein_width, x_y) {
    h1 = -2.8
    h2 = -3.1
    dec = 2 * nchar(name) * protein_width/100
    if (face == "stereoscopic") {
        cylindrect(start, h1, end, h2, col = color, gradient = "y")
    }
    else {
        rect(start, h1, end, h2, col = color)
    }
    if (end - start >= dec) {
        par(srt = 0)
        text((end + start)/2, h1 + height/2, name, cex = 0.7)
        isContain = TRUE
    }
    else {
        isContain = FALSE
    }
    isContain
}
Domain_w = function(domain_pos, domain_name, protein_width) {
    dec = 1.4 * protein_width/100
    position2 = 1:length(domain_pos)
    position2[1] = domain_pos[1]
    if (length(domain_pos) > 1) {
        for (i in 2:length(domain_pos)) {
            if (domain_pos[i] - domain_pos[i - 1] <= dec) {
              if (domain_pos[i] != domain_pos[i - 1]) {
                position2[i] = position2[i - 1] + dec
              }
              else {
                position2[i] = position2[i - 1]
              }
            }
            else {
              position2[i] = domain_pos[i]
            }
        }
    }
    return(position2)
}
Domain_h = function(position, position2, name, height = -0.3,
    x_y, up_down) {
    h1 = -0.1
    h2 = -0.2
    h = -0.4
    hh1 = -2.8
    if (up_down == "up") {
        if (position == position2) {
            segments(position, hh1 + height, position, hh1 +
              height + h)
        }
        else {
            segments(position, hh1 + height, position, hh1 +
              height + h1)
            segments(position2, hh1 + height + h - h2, position2,
              hh1 + height + h)
            segments(position, hh1 + height + h1, position2,
              hh1 + height + h - h2)
        }
        text(position2, hh1 + height + h - 0.02, name, srt = 90,
            adj = c(0, 0.5), cex = 0.8)
    }
    else {
        if (position == position2) {
            segments(position, hh1, position, hh1 - h)
        }
        else {
            segments(position, hh1, position, hh1 - h1)
            segments(position2, hh1 - h + h2, position2,
              hh1 - h)
            segments(position, hh1 - h1, position2, hh1 -
              h + h2)
        }
        text(position2, hh1 - h + 0.02, name, srt = 270,
            adj = c(0, 0.5), cex = 0.8)
    }
}
if (!is.na(domain[1, 1])) {
    domainn = domain
    count = 0
    for (i in 1:nrow(domainn)) {
        isContain = Domain(start = as.numeric(domainn[i,
            1]), end = as.numeric(domainn[i, 2]), name = as.character(domainn[i,
            3]), height = as.numeric(protein[4]), color = i +
            1, face = protein[6], protein_width = as.numeric(length[2]),
            x_y = flag)
        if (isContain == TRUE) {
            domain = domain[-i + count, ]
            count = count + 1
        }
    }
    domain2 = (domain[, 1] + domain[, 2])/2
    if (length(domain2) != 0) {
        flag = TRUE
        if (flag == TRUE) {
            position3 = Domain_w(domain2, domain[, 3], as.numeric(length[2]))
        }
        for (i in 1:nrow(domain)) {
            position1 = (as.numeric(domain[i, 1]) + as.numeric(domain[i,
              2]))/2
            Domain_h(position = position1, position2 = position3[i],
              name = as.character(domain[i, 3]), height = as.numeric(protein[4]),
              x_y = flag, up_down = "down")
        }
        }
      }
    }
```

plotmutagensis *ploting mutagensis*

Description

Draw the mutagensis of the protein

Usage

```
plotmutagensis()
```

Details

The tool ennable visualization of amino acid changes at the protein level,The scale of a protein domain and the position of a functional motif/site will be precisely defined. The features available include mutagensis

Value

The location, height and name of the transition point

Author(s)

Xiaoyu Zhang

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

codehelp

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function ()
{
    protein = read.table("Protein.txt", sep = "\t", stringsAsFactors = F)
    domain = read.table("Domain.txt", sep = "\t", stringsAsFactors = F)
    length = read.table("Length.txt", sep = "\t", stringsAsFactors = F)
```

```
site = read.table("Site.txt", sep = "\t", stringsAsFactors = F)
muta = read.table("Mutagenesis.txt", sep = "\t", stringsAsFactors = F)
option = read.table("Option.txt", sep = "\t", stringsAsFactors = F)
zoomin = read.table("ZoomIn.txt", sep = "\t", stringsAsFactors = F)
Mutagenesis = function(position, position2, color, height2,
    height, up_down, start, end, pc, cex1) {
    h1 = -0.1
    h2 = -1.4
    h = -1.6
    hh1 = -2.8
    if (up_down == "up") {
        if (position == position2) {
            segments(position, hh1 + height, position, hh1 +
              height + h)
        }
        else {
            segments(position, hh1 + height, position, hh1 +
              height + h1)
            segments(position2, hh1 + height + h - h2, position2,
              hh1 + height + h)
            segments(position, hh1 + height + h1, position2,
              hh1 + height + h - h2)
        }
    }
    x = 0
    kong1 = (round(log(start, 10)) + 1) * start/50
    kong2 = (round(log(end, 10)) + 1) * end/50
    if (round(log(end, 10)) + 1 <= 5) {
        kong2 = (round(log(end, 10)) + 1) * end/50
    }
    else {
        kong2 = 5 * end/50
    }
    boxplot(x, xlim = c(start - kong1, end + kong2), ylim = c(1,
        -5.5), axes = FALSE, add = TRUE, border = FALSE)
    points(position2, height2, pch = pc, col = color, cex = cex1)
}
Change_h = function(muta_pos, muta_name, protein_h) {
    d = 0.1
    d1 = 0.26
    hh1 = -2.8
    height2 = 1:length(muta_pos)
    height2[1] = hh1 + protein_h - d1
    position_h = muta_pos
    position_h[1] = muta_pos[1]
    if (length(muta_pos) > 1) {
        for (i in 2:length(muta_pos)) {
            if (muta_pos[i] == position_h[i - 1]) {
              height2[i] = height2[i - 1] - d
            }
            else {
              height2[i] = hh1 + protein_h - d1
            }
```

```
                }
            }
            height2
        }
        Change_m = function(muta, protein_width) {
            dec = 1.4 * protein_width/100
            position3 = 1:length(muta)
            position3[1] = muta[1]
            if (length(muta) > 1) {
                for (i in 2:length(muta)) {
                    if (muta[i] - muta[i - 1] <= dec) {
                      if (muta[i] != muta[i - 1]) {
                        position3[i] = position3[i - 1] + dec
                      }
                      else {
                        position3[i] = position3[i - 1]
                      }
                    }
                    else {
                      position3[i] = muta[i]
                    }
                }
            }
            position3
        }
        if (!is.na(muta[1, 1])) {
            position3 = Change_m(muta[, 1], as.numeric(length[2]))
            height2 = Change_h(muta[, 1], muta[, 2], as.numeric(protein[4]))
            for (i in 1:nrow(muta)) {
                Mutagenesis(position = as.numeric(muta[i, 1]), position2 = position3[i],
                    color = as.character(muta[i, 2]), height2 = height2[i],
                    height = as.numeric(protein[4]), up_down = "up",
                    start = as.numeric(length[1]), end = as.numeric(length[2]),
                    pc = as.numeric(protein[7]), cex1 = as.numeric(protein[8]))
            }
        }
    }
```

plotsite *ploting site*

Description

Draw the protein site

Usage

```
plotsite()
```

Details

The tool ennable visualization of amino acid changes at the protein level,The scale of a protein domain and the position of a functional motif/site will be precisely defined. The features available include site

Value

Location of the site in the protein

Author(s)

Xiaoyu Zhang

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

codehelp

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function ()
{
    protein = read.table("Protein.txt", sep = "\t", stringsAsFactors = F)
    domain = read.table("Domain.txt", sep = "\t", stringsAsFactors = F)
    length = read.table("Length.txt", sep = "\t", stringsAsFactors = F)
    site = read.table("Site.txt", sep = "\t", stringsAsFactors = F)
    muta = read.table("Mutagenesis.txt", sep = "\t", stringsAsFactors = F)
    option = read.table("Option.txt", sep = "\t", stringsAsFactors = F)
    zoomin = read.table("ZoomIn.txt", sep = "\t", stringsAsFactors = F)
    Site = function(position, position2, name, height = -0.3,
        x_y, up_down) {
        h1 = -0.1
        h2 = -0.2
        h = -0.4
        hh1 = -2.8
        if (up_down == "up") {
            if (position == position2) {
                segments(position, hh1 + height, position, hh1 +
                  height + h)
            }
            else {
                segments(position, hh1 + height, position, hh1 +
                  height + h1)
                segments(position2, hh1 + height + h - h2, position2,
                  hh1 + height + h)
                segments(position, hh1 + height + h1, position2,
                  hh1 + height + h - h2)
            }
            text(position2, hh1 + height + h - 0.02, name, srt = 90,
                adj = c(0, 0.5), cex = 0.8)
        }
        else {
            if (position == position2) {
                segments(position, hh1, position, hh1 - h)
            }
            else {
                segments(position, hh1, position, hh1 - h1)
                segments(position2, hh1 - h + h2, position2,
                  hh1 - h)
                segments(position, hh1 - h1, position2, hh1 -
                  h + h2)
            }
            text(position2, hh1 - h + 0.02, name, srt = 270,
                adj = c(0, 0.5), cex = 0.8)
        }
    }
    Change_x = function(site_pos, site_name, protein_width) {
        dec = 1.4 * protein_width/100
        position2 = 1:length(site_pos)
        position2[1] = site_pos[1]
        if (length(site_pos) > 1) {
            for (i in 2:length(site_pos)) {
                if (site_pos[i] - site_pos[i - 1] <= dec) {
                  if (site_pos[i] != site_pos[i - 1]) {
                    position2[i] = position2[i - 1] + dec
                  }
                  else {
                    position2[i] = position2[i - 1]
                  }
                }
                else {
                  position2[i] = site_pos[i]
                }
            }
        }
        return(position2)
    }
    if (!is.na(site[1, 1])) {
        position2 = Change_x(site[, 1], site[, 2], as.numeric(length[2]))
        for (i in 1:nrow(site)) {
            Site(position = as.numeric(site[i, 1]), position2 = position2[i],
                name = as.character(site[i, 2]), height = as.numeric(protein[4]),
                x_y = flag, up_down = "up")
        }
    }
}
```

site_data *downloading protein site*

Description

Download the site of the protein, including the name

Usage

```
site_data()
```

Details

Download the site of the protein, including the distribution of the locus of the marker space

Value

The location of the marker line

Author(s)

Xiaoyu Zhang

References

https://cran.r-project.org/doc/manuals/R-exts.html

See Also

codehelp

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function ()
{
    library(XML)
    library(plyr)
    protein = read.table("Protein.txt", sep = "\t", stringsAsFactors = F)
    name = protein[2]
    url_p = "http://www.uniprot.org/uniprot/"
    url_s = "#showFeatures"
    url_w = paste(url_p, name, url_s, sep = "")
    url = url_w
    doc <- htmlParse(url)
 position_s = xpathSApply (doc, "//table[@id= 'sitesAnno_section']
  /tr/td/  a[@class = 'position tooltipped']",
        xmlValue)
   name_s = xpathSApply (doc, "//table[@id= 'sitesAnno_section']/tr/td/span[@property='text']",
        xmlValue)
    s_s <- c()
    for (i in 1:length(position_s)) {
        s_s[i] <- gsub(pattern = "//D", replacement = "x", position_s[i])
    }
    s_s <- strsplit(s_s, "xxx")
    d1_s <- laply(s_s, function(x) x[1])
    d2_s <- laply(s_s, function(x) x[2])
    r1_site = d1_s
    r2_site = name_s
    dfrm_site = data.frame(r1_site, r2_site)
    write.table(dfrm_site, file = "Site.txt", sep = "/t", quote = FALSE,
        row.names = F, col.names = F)
  }
```

Index