# Package 'ProgModule'

January 20, 2025

**Type** Package

**Title** Identification of Prognosis-Related Mutually Exclusive Modules

**Version** 0.1.0

**Maintainer** Junwei Han <hanjunwei1981@163.com>

**Description** A novel tool to identify candidate driver modules for predicting the prognosis of patients by integrating exclusive coverage of mutations with clinical characteristics in cancer.

**License** GPL (>= 2)

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 4.2.0)

**RoxygenNote** 7.2.3

**Imports** igraph, infotheo, maftools, patchwork, pathwayTMB, stats, survival, survminer, utils

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Junwei Han [aut, cre, cph],
Xiangmei Li [aut],
Bingyue Pan [aut]

**Repository** CRAN

**Date/Publication** 2024-05-16 14:40:05 UTC

## Contents

---

`candidate_module` *candidate_module, candidate module*

---

## Description

candidate_module, Is the gene set from each local network by greedy algorithm,generated by 'get_candidate_module'.

## Usage

```
candidate_module
```

## Format

An object of class `list` of length 4.

---

`final_candidate_module`
*final_candidate_module, Final candidate modules*

---

## Description

final_candidate_module, Is the final candidate module by intersecting modules from three protein networks in pairs,generated by 'get_final_candidate_module'.

### Usage

```
final_candidate_module
```

### Format

An object of class `list` of length 4.

---

`get_candidate_module` *Get candidate module.*

---

### Description

The function 'get_candidate_module' is used to search candidate module of each local network using greedy algorithm.

### Usage

```
get_candidate_module(
  local_network,
  network,
  freq_matrix,
  sur,
  seed,
  max.size,
  rate
)
```

### Arguments

| | |
|---|---|
| `local_network` | The local networks,generated by 'get_local_network'. |
| `network` | The maximum connected subnet,extracted by mapping all mutated genes to the PPI network. |
| `freq_matrix` | The mutations matrix,generated by 'get_mut_status'. |
| `sur` | A nx2 data frame of samples' survival data,the first line is samples' survival event and the second line is samples' overall survival. |
| `seed` | The canonical drivers from NCG database, which use as the starting node of the greedy algorithm. |
| `max.size` | The maximum size of the module,default is 200. |
| `rate` | The rate of increase in score,default is 0.05. |

### Value

candidate module.

## Examples

```
#load the data
data(local_network)
data(mut_status)
data(subnet)
canonical_drivers<-system.file("extdata","canonical_drivers.txt",package = "ProgModule")
seed_gene<-read.table(canonical_drivers,header=FALSE)
sur<-system.file("extdata","sur.csv",package = "ProgModule")
sur<-read.delim(sur,sep=",",header=TRUE,row.names=1)
#perform the function `get_candidate_module`.
candidatemodule.example<-get_candidate_module(local_network=local_network,network=subnet,
freq_matrix=mut_status,sur=sur,seed=seed_gene[,1],max.size=200,rate=0.05)
```

---

`get_final_module` *Get final module.*

---

## Description

The function 'get_final_module' is used to identify the final module.

## Usage

```
get_final_module(
  index,
  edge,
  mut_status,
  sur,
  seed,
  cutoff = 0.05,
  max.size.local = 500,
  max.size.candidate = 200,
  rate = 0.05,
  perm = 1000
)
```

## Arguments

| | |
|---|---|
| `index` | A index file of PPI networks,downloaded from http://compbio-research.cs.brown.edu/pancancer/hotnet2/. |
| `edge` | The edge lists of PPI networks,downloaded from http://compbio-research.cs.brown.edu/pancancer/hotnet2 |
| `mut_status` | The mutations matrix,generated by 'get_mut_status'. |
| `sur` | A nx2 data frame of samples' survival data,the first line is samples' survival event and the second line is samples' overall survival. |
| `seed` | The canonical drivers from NCG database, which use as the starting node of the greedy algorithm. |
| `cutoff` | The perturbed p-value cutoff point,default is 0.05. |
| `max.size.local` | The size of maximum connected local network,default is 500. |

max.size.candidate
: The maximum size of the candidate module,default is 200.

rate
: The rate of increase in score,default is 0.05.

perm
: The perturbation number,default is 1000.

## Value

The final module.

## Examples

```
#load the data
indexdata<-system.file("extdata","hint+hi2012_index_file.txt",package="ProgModule")
index<-read.table(indexdata,sep="\t",header=FALSE)
edgedata<-system.file("extdata","hint+hi2012_edge_file.txt",package="ProgModule")
edge<-read.table(edgedata,sep="\t",header=FALSE)
data(mut_status)
sur<-system.file("extdata","sur.csv",package ="ProgModule")
sur<-read.delim(sur,sep=",",header=TRUE,row.names=1)
canonical_drivers<-system.file("extdata","canonical_drivers.txt",package="ProgModule")
seed_gene<-read.table(canonical_drivers,header=FALSE)
#perform the function `get_final_module`.
finalmodule.example<-get_final_module(index,edge,mut_status,sur,seed=seed_gene,
cutoff=0.05,max.size.local=500,max.size.candidate=200,rate=0.05,perm=100)
```

---

# get_local_network *Extract the local networks from the PPI network.*

---

## Description

The function 'get_local_network' is used to search local network of each gene by breadth-first algorithm.

## Usage

```
get_local_network(network, freq_matrix, max.size = 500)
```

## Arguments

network
: The PPI network.

freq_matrix
: The mutations matrix,generated by 'get_mut_status'.

max.size
: The size of maximum connected local network,default is 500.

## Value

local nerwork.

## Examples

```
#load the data
data(mut_status)
data(subnet)
#perform the function `get_local_network`.
localnetwork.example<-get_local_network(network=subnet,freq_matrix=mut_status,max.size=500)
```

---

`get_lollipopPlot` *Draw an lollipopPlot for module genes*

---

## Description

Load the data in MAF format and draws an lollipopPlot.

## Usage

```
get_lollipopPlot(
  maf,
  gene,
  AACol = NULL,
  labelPos = NULL,
  labPosSize = 0.9,
  showMutationRate = TRUE,
  showDomainLabel = TRUE,
  cBioPortal = FALSE,
  refSeqID = NULL,
  proteinID = NULL,
  roundedRect = TRUE,
  repel = FALSE,
  collapsePosLabel = TRUE,
  showLegend = TRUE,
  legendTxtSize = 0.8,
  labPosAngle = 0,
  domainLabelSize = 0.8,
  axisTextSize = c(1, 1),
  printCount = FALSE,
  colors = NULL,
  domainAlpha = 1,
  domainBorderCol = "black",
  bgBorderCol = "black",
  labelOnlyUniqueDoamins = TRUE,
  defaultYaxis = FALSE,
  titleSize = c(1.2, 1),
  pointSize = 1.5
)
```

### Arguments

| | |
|---|---|
| maf | The patients' somatic mutation data, which in MAF format. |
| gene | Modular gene from final_candidate_module,generated by 'get_final_candidate_module'. |
| AACol, labelPos, labPosSize, showMutationRate, showDomainLabel, cBioPortal, refSeqID, proteinID, roundedRect, repel, collapsePosLabel, showLegend, legendTxtSize, labPosAngle, domainLabelSize, axisTextSize, printCount, colors, domainAlpha, domainBorderCol, bgBorderCol, labelOnlyUniqueDoamins, defaultYaxis, titleSize, pointSize | see lollipopPlot |

### Value

No return value

### Examples

```
#load the data.
maffile<-system.file("extdata","maffile.maf",package="ProgModule")
#draw an lollipopPlot
get_lollipopPlot(maf=maffile,gene="TP53")
```

---

## get_mutual_module *Extract the mutually exclusive module.*

---

### Description

The function 'get_mutual_module' is used to determine if neighbor genes should be added to the module by calculating the score.

### Usage

```
get_mutual_module(
  module,
  net,
  freq_matrix,
  sur,
  module_sig,
  univarCox_result,
  rate
)
```

### Arguments

| | |
|---|---|
| module | The Original modular gene set. |
| net | The local network extracted from PPI network. |
| freq_matrix | The mutations matrix,generated by 'get_mut_status'. |

| | |
|---|---|
| sur | A nx2 data frame of samples' survival data,the first line is samples' survival event and the second line is samples' overall survival. |
| module_sig | A label for whether the module is a risk factor or a protective factor for survival. |
| univarCox_result | The result of Cox univariate analysis,generated by 'get_univarCox_result'. |
| rate | The rate of increase in score,default is 0.05. |

## Value

The mutually exclusive module.

## Examples

```
#load the data
data(mut_status)
sur<-system.file("extdata","sur.csv",package = "ProgModule")
sur<-read.delim(sur,sep=",",header=TRUE,row.names = 1)
data(net)
data(module)
data(univarCox_result)
#perform the function `get_mutual_module`.
mutuallyexclusivemodule.example<-get_mutual_module(module,net,freq_matrix=mut_status,sur,
module_sig="risk",univarCox_result,rate=0.05)
```

---

# get_mut_status *Converts MAF file into mutation matrix.*

---

## Description

The function 'get_mut_status' uses to convert MAF file into mutation matrix.

## Usage

```
get_mut_status(mutvariant, nonsynonymous = TRUE)
```

## Arguments

| | |
|---|---|
| mutvariant | A nx3 data frame of patients' somatic mutation data,the first line is gene symbol,the second line is sample ID and the third line is mutation classification. |
| nonsynonymous | Logical, tell if extract the non-synonymous somatic mutations (nonsense mutation, missense mutation, frame-shif indels, splice site, nonstop mutation, translation start site, inframe indels). |

## Value

A binary mutations matrix, in which 1 represents that a particular gene has mutated in a particular sample, and 0 represents that gene has no mutation in a particular sample.

## Examples

```
maf<-system.file("extdata","maffile.maf",package = "ProgModule")
maf_data<-read.delim(maf)
mutvariant<-maf_data[,c("Hugo_Symbol","Tumor_Sample_Barcode","Variant_Classification")]
#perform the function `get_mut_status`.
mut_status.example<-get_mut_status(mutvariant,nonsynonymous = TRUE)
```

---

`get_mut_survivalresult` *Plot Kaplan-Meier survival curve.*

---

## Description

The function 'get_mut_survivalresult' uses to draw the Kaplan-Meier survival curve based on the mutated status of candidate module.

## Usage

```
get_mut_survivalresult(module, freq_matrix, sur)
```

## Arguments

| | |
|---|---|
| `module` | The gene module,generated by 'get_final_candidate_module'. |
| `freq_matrix` | The mutations matrix,generated by 'get_mut_status'. |
| `sur` | A nx2 data frame of samples' survival data,the first line is samples' survival event and the second line is samples' overall survival. |

## Value

No return value

## Examples

```
#load the data.
data(mut_status)
sur<-system.file("extdata","sur.csv",package="ProgModule")
sur<-read.delim(sur,sep=",",header=TRUE,row.names=1)
data(final_candidate_module)
#perform the function `get_mut_survivalresult`.
get_mut_survivalresult(module=final_candidate_module,freq_matrix=mut_status,sur)
```

get_oncoplots *Draw a waterfall plot of mutated genes involved in the module*

## Description

Load the data in MAF format and draws a waterfall plot of mutated genes involved in the module.

## Usage

```
get_oncoplots(
  maf,
  genes,
  removeNonMutated = TRUE,
  top = 20,
  minMut = NULL,
  altered = FALSE,
  drawRowBar = TRUE,
  drawColBar = TRUE,
  leftBarData = NULL,
  leftBarLims = NULL,
  rightBarData = NULL,
  rightBarLims = NULL,
  topBarData = NULL,
  logColBar = FALSE,
  includeColBarCN = TRUE,
  clinicalFeatures = NULL,
  annotationColor = NULL,
  annotationDat = NULL,
  pathways = NULL,
  path_order = NULL,
  selectedPathways = NULL,
  pwLineCol = "#535c68",
  pwLineWd = 1,
  draw_titv = FALSE,
  titv_col = NULL,
  showTumorSampleBarcodes = FALSE,
  barcode_mar = 4,
  barcodeSrt = 90,
  gene_mar = 5,
  anno_height = 1,
  legend_height = 4,
  sortByAnnotation = FALSE,
  groupAnnotationBySize = TRUE,
  annotationOrder = NULL,
  sortByMutation = FALSE,
  keepGeneOrder = FALSE,
  GeneOrderSort = TRUE,
```

```
  sampleOrder = NULL,
  additionalFeature = NULL,
  additionalFeaturePch = 20,
  additionalFeatureCol = "gray70",
  additionalFeatureCex = 0.9,
  genesToIgnore = NULL,
  fill = TRUE,
  cohortSize = NULL,
  colors = NULL,
  cBioPortal = FALSE,
  bgCol = "#CCCCCC",
  borderCol = "white",
  annoBorderCol = NA,
  numericAnnoCol = NULL,
  drawBox = FALSE,
  fontSize = 0.8,
  SampleNamefontSize = 1,
  titleFontSize = 1.5,
  legendFontSize = 1.2,
  annotationFontSize = 1.2,
  sepwd_genes = 0.5,
  sepwd_samples = 0.25,
  writeMatrix = FALSE,
  colbar_pathway = FALSE,
  showTitle = TRUE,
  titleText = NULL,
  showPct = TRUE
)
```

## Arguments

| | |
|---|---|
| maf | The patients' somatic mutation data, which in MAF format. |
| genes | Modular gene set from final_candidate_module,generated by 'get_final_candidate_module'. |
| removeNonMutated, top, minMut, altered, drawRowBar, drawColBar, leftBarData, leftBarLims, rightBarData, rightBarLims, topBarData, logColBar, includeColBarCN, clinicalFeatures, annotationColor, annotationDat, pathways, path_order, selectedPathways, pwLineCol, pwLineWd, draw_titv, titv_col, showTumorSampleBarcodes, barcode_mar, barcodeSrt, gene_mar, anno_height, legend_height, sortByAnnotation, groupAnnotationBySize, annotationOrder, sortByMutation, keepGeneOrder, GeneOrderSort, sampleOrder, additionalFeature, additionalFeaturePch, additionalFeatureCol, additionalFeatureCex, genesToIgnore, fill, cohortSize, colors, cBioPortal, bgCol, borderCol, annoBorderCol, numericAnnoCol, drawBox, fontSize, SampleNamefontSize, titleFontSize, legendFontSize, annotationFontSize, sepwd_genes, sepwd_samples, writeMatrix, colbar_pathway, showTitle, titleText, showPct | see oncoplot |

## Value

No return value

## Examples

```
#load the data.
maffile<-system.file("extdata","maffile.maf",package="ProgModule")
data(final_candidate_module)
#draw an oncoplot
get_oncoplots(maf=maffile,genes=final_candidate_module[[1]])
```

---

# get_plotMutInteract *Exact tests to detect mutually exclusive, co-occuring and altered gene-sets or pathways.*

---

## Description

Performs Pair-wise Fisher's Exact test to detect mutually exclusive or co-occuring events.

## Usage

```
get_plotMutInteract(
  module = NULL,
  genes = NULL,
  freq_matrix,
  pvalue = c(0.05, 0.01),
  returnAll = TRUE,
  fontSize = 0.8,
  showSigSymbols = TRUE,
  showCounts = FALSE,
  countStats = "all",
  countType = "all",
  countsFontSize = 0.8,
  countsFontColor = "black",
  colPal = "BrBG",
  nShiftSymbols = 5,
  sigSymbolsSize = 2,
  sigSymbolsFontSize = 0.9,
  pvSymbols = c(46, 42),
  limitColorBreaks = TRUE
)
```

## Arguments

| | |
|---|---|
| module | The gene module,generated by 'get_final_candidate_module'. |
| genes | The modular gene,generated by 'get_final_candidate_module'. |
| freq_matrix | The mutations matrix,generated by 'get_mut_status'. |

pvalue, returnAll, fontSize, showSigSymbols, showCounts, countStats, countType, countsFontSize, countsFontColor, colPal, nShiftSymbols, sigSymbolsSize, sigSymbolsFontSize, pvSymbols, limitColorBreaks
see plotMutInteract

## Value

No return value

## Examples

```
#load the data.
data(plotMutInteract_moduledata,plotMutInteract_mutdata)
#draw an plotMutInteract of genes
get_plotMutInteract(genes=unique(unlist(plotMutInteract_moduledata)),
freq_matrix=plotMutInteract_mutdata)
#draw an plotMutInteract of modules
get_plotMutInteract(module=plotMutInteract_moduledata,
freq_matrix=plotMutInteract_mutdata)
```

---

# get_Pvalue_S *Get perturbed p-value.*

---

## Description

The function 'get_Pvalue_S' is used to calculate the perturbed p-value.

## Usage

```
get_Pvalue_S(module, freq_matrix, sur, perms = 1000, local_network)
```

## Arguments

| | |
|---|---|
| module | The candidate module,generated by 'get_candidate_module'. |
| freq_matrix | The mutations matrix,generated by 'get_mut_status'. |
| sur | A nx2 data frame of samples' survival data,the first line is samples' survival event and the second line is samples' overall survival. |
| perms | The perturbation number,default is 1000. |
| local_network | The local network gene sets,generated by 'get_local_network'. |

## Value

Perturbed p-value.

## Examples

```
#load the data
data(local_network)
data(mut_status)
data(candidate_module)
sur<-system.file("extdata","sur.csv",package = "ProgModule")
sur<-read.delim(sur,sep=",",header=TRUE,row.names = 1)
#perform the function `get_Pvalue_S`.
turbulence.example<-get_Pvalue_S(module=candidate_module,freq_matrix=mut_status,
sur=sur,perms=100,local_network)
```

---

# get_univarCox_result *Get univarCox result.*

---

## Description

The function 'get_univarCox_result' is used to calculate the result of Cox univariate analysis.

## Usage

```
get_univarCox_result(freq_matrix, sur)
```

## Arguments

| | |
|---|---|
| freq_matrix | The mutations matrix,generated by 'get_mut_status'. |
| sur | A nx2 data frame of samples' survival data,the first line is samples' survival event and the second line is samples' overall survival. |

## Value

The result of Cox univariate analysis.

## Examples

```
#load the data
data(mut_status)
sur<-system.file("extdata","sur.csv",package ="ProgModule")
sur<-read.delim(sur,sep=",",header=TRUE,row.names=1)
#perform the function `get_univarCox_result`.
univarCoxresult.example<-get_univarCox_result(freq_matrix=mut_status,sur)
```

## local_network *local_network, local network gene set*

### Description

local_network, the local network gene set of each gene by breadth-first algorithm,,generated by ‘get_local_network‘.

### Usage

```
local_network
```

### Format

An object of class `list` of length 4.

## maf_data *maf_data, MAF file*

### Description

maf_data, The patients’ somatic mutation data, which in MAF format.

### Usage

```
maf_data
```

### Format

An object of class `data.frame` with 3745 rows and 10 columns.

## MI *Calculate mutual information.*

### Description

The function ‘MI‘ is used to calculate the mutual information score between samples’ survival status and mutation status.

### Usage

```
MI(mylist1, mylist2)
```

### Arguments

| | |
|---|---|
| mylist1 | Is input a list of samples' survival status. |
| mylist2 | Is input a list of samples' mutation status. |

### Value

The mutual information score

### Examples

```
#load the data
data(mut_status)
sur<-system.file("extdata","sur.csv",package = "ProgModule")
sur<-read.delim(sur,sep=",",header=TRUE,row.names = 1)
#perform the function 'MI'
mut_matrix<-MI(mylist1 = as.numeric(mut_status[1,]),mylist2 = sur[,1])
```

---

## module *module, gene set*

---

### Description

module, Original modular gene set.

### Usage

```
module
```

### Format

An object of class `character` of length 1.

---

## mut_status *mut_status, mutations matrix*

---

### Description

mut_status, the mutations matrix,generated by 'get_mut_status'.

### Usage

```
mut_status
```

### Format

An object of class `matrix` (inherits from `array`) with 338 rows and 331 columns.

`net` *net, network*

## Description

net, Is a local network extracted from the ppi network.

## Usage

```
net
```

## Format

An object of class `igraph` of length 76.

`plotMutInteract_moduledata` *plotMutInteract_moduledata*

## Description

The data use for drawing mutually exclusive and co-occurrence plots.

## Usage

```
plotMutInteract_moduledata
```

## Format

An object of class `list` of length 7.

`plotMutInteract_mutdata` *plotMutInteract_mutdata*

## Description

The data use for drawing mutually exclusive and co-occurrence plots.

## Usage

```
plotMutInteract_mutdata
```

## Format

An object of class `matrix` (inherits from `array`) with 89 rows and 430 columns.

subnet *subnet, network*

## Description

subnet, Is a maximum connected subnet,extracted by mapping all genes to the ppi network.

## Usage

```
subnet
```

## Format

An object of class igraph of length 1624.

univarCox_result *univarCox_result*

## Description

The result of Cox univariate analysis,generated by ‘get_univarCox_result‘.

## Usage

```
univarCox_result
```

## Format

An object of class numeric of length 8103.

# Index