Package ‘Cascade’
August 24, 2019

Type Package
Title Selection, Reverse-Engineering and Prediction in Cascade Networks
Version 1.8
Date 2019-08-24
Depends R (>= 3.5.0)

biocViews
Imports abind, animation, cluster, grid, igraph, lars, lattice, limma,
magic, methods, nlns, splines, stats4, survival, tnet, VGAM
Suggests R.rsp, CascadeData, knitr

Author Frederic Bertrand [cre, aut] (<https://orcid.org/0000-0002-0837-8281>),
Myriam Maumy-Bertrand [aut] (<https://orcid.org/0000-0002-4615-1512>),
Laurent Vallat [ctb],
Nicolas Jung [ctb]

Maintainer Frederic Bertrand <frederic.bertrand@math.unistra.fr>

Description A modeling tool allowing gene selection, reverse engineering, and prediction in cascade networks.

License GPL (>= 2)
Encoding UTF-8

micropredict.R

Classification/MSC 62J05, 62J07, 62J99, 92C42
VignetteBuilder R.rsp

RoxygenNote 6.1.1

URL http://www-irma.u-strasbg.fr/~fbertran/,
https://github.com/fbertran/Cascade

BugReports https://github.com/fbertran/Cascade/issues
NeedsCompilation no
**Cascade-package**

**Repository** CRAN

**Date/Publication** 2019-08-24 04:20:06 UTC

### R topics documented:

- Cascade-package .................................................. 2
- analyze_network .................................................... 3
- as.micro_array .................................................... 4
- compare-methods ................................................... 5
- cutoff ............................................................... 6
- dim ................................................................. 7
- evolution ............................................................ 7
- geneNeighborhood .................................................. 8
- genePeakSelection .................................................. 9
- gene_expr_simulation .............................................. 12
- head ................................................................. 13
- inference ............................................................ 14
- M ..................................................................... 15
- micropredict-class ............................................... 15
- micro_array-class ................................................ 16
- Net ................................................................. 17
- network ............................................................. 17
- network-class ...................................................... 18
- network_random .................................................... 19
- Net_inf ............................................................... 20
- plot-methods ....................................................... 20
- position-methods .................................................. 21
- predict ............................................................. 22
- print-methods ....................................................... 22
- Selection ........................................................... 23
- summary-methods ................................................... 23
- unionMicro-methods ............................................... 24

**Index**

- Cascade-package ............................... The Cascade Package

### Description

A modeling tool allowing gene selection, reverse engineering, and prediction in Cascade networks.
analyze_network

Details

Package: Cascade
Type: Package
Version: 1.7
Date: 2019-02-09
License: GNU 2.0
Depends: methods

Author(s)

This package has been written by Frédéric Bertrand, Myriam Maumy-Bertrand and Nicolas Jung with biological insights from Laurent Vallat. Maintainer: Frédéric Bertrand <frederic.bertrand@math.unistra.fr>

References


analyze_network    Analysing the network

Description

Calculates some indicators for each node in the network.

Usage

analyze_network(Omega,nv,...)

Arguments

Omega       a network object
nv         the level of cutoff at which the analysis should be done
...        label_v : (optionnal) name of the genes

Value

A matrix containing, for each node, its betweenness, its degree, its output, its closeness.
Author(s)
Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.

References


Examples
```r
data(network)
analyze_network(network, nv=0)
```

---

**as.micro_array**

Coerce a matrix into a micro_array object.

Description
Coerce a matrix into a micro_array object.

Usage
```r
as.micro_array(M, time, subject)
```

Arguments
- **M**  
  A matrix. Contains the microarray measurements. Should be of size N * K, with N the number of genes and K=T*P with T the number of time points, and P the number of individuals. This matrix should be created using `cbind(M1,M2,...)` with M1 a N*T matrix with the measurements for individual 1, M2 a N*T matrix with the measurements for individual 2.
- **time**  
  A vector. The time points measurements.
- **subject**  
  The number of subjects.

Value
A micro_array object.

Author(s)
Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.
compare-methods

References


Examples
if(require(CascadeData)){
  data(micro_US)
  micro_US<-as.micro_array(micro_US,time=c(60,90,210,390),subject=6)
}

compare-methods Some basic criteria of comparison between actual and inferred network.

Description
Allows comparison between actual and inferred network.

Value
A vector containing: sensibility, predictive positive value, and the F-score

Methods
signature(Net = "network", Net_inf = "network", nv = "numeric") Net A network object containing the actual network.

Net_inf A network object containing the inferred network.

nv A number that indicates at which level of cutoff the comparison should be done.

Author(s)
Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.

References

Examples

```r
data(Net)
data(Net_inf)

#Comparing true and inferred networks
F_score=NULL

#Here are the cutoff level tested
test.seq<-seq(0,max(abs(Net_inf@network*0.9)),length.out=200)
for(u in test.seq){
  F_score<-rbind(F_score,Cascade::compare(Net,Net_inf,u))
}
matplot(test.seq,F_score,type="l",ylab="criterion value",xlab="cutoff level",lwd=2)
```

cutoff

Choose the best cutoff

Description

Allows estimating the best cutoff, in function of the scale-freeness of the network. For a sequence of cutoff, the corresponding p-value is then calculated.

Usage

cutoff(Omega,...)

Arguments

Omega a network object

... Optional arguments:

  sequence a vector corresponding to the sequence of cutoffs that will be tested.
  x_min an integer ; only values over x_min are further retained for performing the test.

Value

A list containing two objects :

  p.value the p values corresponding to the sequence of cutoff
  p.value.inter the smoothed p value vector, using the loess function

Author(s)

Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.
References


Examples

data(network)
cutoff(network)
#See vignette for more details

dim

*Dimension of the data*

Description

Dimension of the data

Methods

signature(x = "micro_array") Gives the dimension of the matrix of measurements.

Examples

if(require(CascadeData)){
data(micro_US)
micro_US<-as.micro_array(micro_US,time=c(60,90,210,390),subject=6)
dim(micro_US)
}

evolution

*See the evolution of the network with change of cutoff*

Description

See the evolution of the network with change of cutoff. This function may be usefull to see if the global topology is changed while increasing the cutoff.

Usage

evolution(net,list_nv,...)
Arguments

net  a network object
list_nv a vector of cutoff at which the network should be shown
...

Optionnal arguments:
gr  a vector giving the group of each gene
color.vertex  a vector giving the color of each node
fix  logical, should the position of the node in the network be calculated once at
     the beginning ? Defaut to TRUE.
taille  vector giving the size of the plot. Default to c(2000,1000)
...

see plot function

Value

A HTML page with the evolution of the network.

Author(s)

Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.

References


Examples

data(network)
sequence<-seq(0,0.2,length.out=20)
#setwd("inst/animation")
#evolution(network,sequence)

geneNeighborhood  Find the neighborhood of a set of nodes.

Description

Find the neighborhood of a set of nodes.

Usage

geneNeighborhood(net,targets,...)
genePeakSelection

Arguments

net a network object
targets a vector containing the set of nodes
... Optional arguments. See plot options.

Value

The neighborhood of the targeted genes.

Author(s)

Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.

References


Examples

data(Selection)
data(network)
#A nv value can chosen using the cutoff function
nv=.11
EGR1<-which(match(Selection@name,"EGR1")==1)
P<-position(network,nv=nv)
geneNeighborhood(network,targets=EGR1,nv=nv,ini=P,
label_v=network@name)

genePeakSelection Methods for selecting genes

Description

Selection of differentially expressed genes.

Usage

geneSelection(x,y,tot.number,...)
genePeakSelection(x,peak,...)
genePeakSelection

Arguments

x  either a micro_array object or a list of micro_array objects. In the first case, the micro_array object represents the stimulated measurements. In the second case, the control unstimulated data (if present) should be the first element of the list.

y  either a micro_array object or a list of strings. In the first case, the micro_array object represents the stimulated measurements. In the second case, the list is the way to specify the contrast:

First element:  condition, condition&time or pattern. The condition specification is used when the overall is to compare two conditions. The condition&time specification is used when comparing two conditions at two precise time points. The pattern specification allows to decide which time point should be differentially expressed.

Second element:  a vector of length 2. The two conditions which should be compared. If a condition is used as control, it should be the first element of the vector. However, if this control is not measured through time, the option cont=TRUE should be used.

Third element:  depends on the first element. It is no needed if condition has been specified. If condition&time has been specified, then this is a vector containing the time point at which the comparison should be done. If pattern has been specified, then this is a vector of 0 and 1 of length T, where T is the number of time points. The time points with desired differential expression are provided with 1.

tot.number  an integer. The number of selected genes. If tot.number <0 all differentially genes are selected. If tot.number > 1, tot.number is the maximum of differentially genes that will be selected. If 0<tot.number<1, tot.number represents the proportion of differentially genes that are selected.

peak  interger. At which time points measurements should the genes be selected [optional for geneSelection].

...  Optional arguments:

M2  a micro_array object. The unstimulated measurements.

data_log  logical (default to TRUE) ; should data be logged ?

wanted.patterns  a matrix with wanted patterns [only for geneSelection].

forbidden.patterns  a matrix with forbidden patterns [only for geneSelection].

durPeak  vector of size 2 (default to c(1,1)) ; the first elements gives the length of the peak at the left, the second at the right. [only for genePeakSelection]

abs_val  logical (default to TRUE) ; should genes be selected on the basis of their absolute value expression ? [only for genePeakSelection]

alpha_diff  float ; the risk level

Value

A micro_array object.

Author(s)

Nicolas Jung, Frédéric Bertrand , Myriam Maumy-Bertrand.
References


Examples

```r
if(require(CascadeData)){
  data(micro_US)
  micro_US<-as.micro_array(micro_US,time=c(60,90,210,390),subject=6)
  data(micro_S)
  micro_S<-as.micro_array(micro_S,time=c(60,90,210,390),subject=6)

  #Basically, to find the 50 more significant expressed genes you will use:
  Selection_1<-geneSelection(x=micro_S,y=micro_US,
    tot.number=50,data_log=TRUE)
  summary(Selection_1)

  #If we want to select genes that are differentially
  #at time t60 or t90 :
  Selection_2<-geneSelection(x=micro_S,y=micro_US,tot.number=30,
    wanted.patterns= rbind(c(0,1,0,0),c(1,0,0,0),c(1,1,0,0)))
  summary(Selection_2)

  #To select genes that have a differential maximum of expression at a specific time point.
  Selection_3<-genePeakSelection(x=micro_S,y=micro_US,peak=1,
    abs_val=FALSE,alpha_diff=0.01)
  summary(Selection_3)
}
```

```r
if(require(CascadeData)){
  data(micro_US)
  micro_US<-as.micro_array(micro_US,time=c(60,90,210,390),subject=6)
  data(micro_S)
  micro_S<-as.micro_array(micro_S,time=c(60,90,210,390),subject=6)

  #Genes with differential expression at t1
  Selection1<-geneSelection(x=micro_S,y=micro_US,20,wanted.patterns= rbind(c(1,0,0,0)))
  #Genes with differential expression at t2
  Selection2<-geneSelection(x=micro_S,y=micro_US,20,wanted.patterns= rbind(c(0,1,0,0)))
  #Genes with differential expression at t3
  Selection3<-geneSelection(x=micro_S,y=micro_US,20,wanted.patterns= rbind(c(0,0,1,0)))
  #Genes with differential expression at t4
  Selection4<-geneSelection(x=micro_S,y=micro_US,20,wanted.patterns= rbind(c(0,0,0,1)))
  #Genes with global differential expression
  Selection5<-geneSelection(x=micro_S,y=micro_US,20)
}
We then merge these selections:
Selection<-unionMicro(list(Selection1, Selection2, Selection3, Selection4, Selection5))
print(Selection)

# Prints the correlation graphics Figure 4:
summary(Selection, 3)

## Uncomment this code to retrieve gene ids.
library(org.Hs.eg.db)
#
# ff<-function(x){substr(x, 1, nchar(x)-3)}
# ff<-Vectorize(ff)
#
## Here is the function to transform the probeset names to gene ID.
#
# library("hgu133plus2.db")
#
# probe_to_id<-function(n){
# x <- hgu133plus2SYMBOL
# mp<-mappedkeys(x)
# xx <- unlist(as.list(x[mp]))
# genes_all = xx[(n)]
# genes_all[is.na(genes_all)]<="unknown"
# return(genes_all)
#
# Selection@name<-probe_to_id(Selection@name)
}

---

gene_expr_simulation  Simulates microarray data based on a given network.

Description
Simulates microarray data based on a given network.

Usage
gene_expr_simulation(network,...)

Arguments

network  A network object.
...
   time_label  a vector containing the time labels.
   subject  the number of subjects
   level_peak  the mean level of peaks.
Value

A micro_array object.

Author(s)

Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.

References


Examples

data(Net)
set.seed(1)

# We simulate gene expression according to the network Net
Msim<-gene_expr_simulation(
  network=Net,  
time_label=rep(1:4,each=25),  
subject=5,  
level_peak=200)
head(Msim)
inference

Reverse-engineer the network

Description
Reverse-engineer the network.

Usage
inference(M,...)

Arguments

M a micro_array object.

Optional arguments:

- tour.max=30 maximal number of steps.
- g=function(x) 1/x the new solution is choosen as (the old solution + g(x) * the
  new solution)/(1+g(x)) where x is the number of steps.
- conv=10e-3 convergence criterion.
- cv.subjects=TRUE should the cross validation be done removing the subject
  one by one ?
- nb.folds=NULL Relevant only if cv.subjects is FALSE. The number of folds in
  cross validation.
- eps=10e-5 machine zero
- type.inf="iterative" "iterative" or "noniterative" : should the algorithm be com-
  puted iteratively

Value
A network object.

Author(s)
Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.

References
package to study, predict and simulate the diffusion of a signal through a temporal gene network. Bioinformatics, btt705.

Examples

# With simulated data
data(M)
infM <- inference(M)
str(infM)

# With selection of genes from GSE39411
data(Selection)
infSel <- inference(Selection)
str(infSel)

M
Simulated M data for examples.

Description
Simulated M microarray.

Usage
data(M)

Examples
data(M)
head(M)

micropredict-class
Class "micropredict"

Description
Class for prediction of microarray value.

Objects from the Class
Objects can be created by calls of the form new("micropredict", ...).

Examples
showClass("micropredict")
Description
The Class

Objects from the Class
Objects can be created by calls of the form new("micro_array", ...).

Slots
microarray: Object of class "matrix" ~
name: Object of class "vector" ~
group: Object of class "vector" ~
start_time: Object of class "vector" ~
time: Object of class "vector" ~
subject: Object of class "numeric" ~

Methods

dim signature(x = "micro_array"): ...
genePeakSelection signature(M1 = "micro_array", M2 = "micro_array", peak = "numeric"): ...
geneSelection signature(x = "micro_array", y = "micro_array", tot.number = "numeric"): ...
geneSelection signature(x = "list", y = "list", tot.number = "numeric"): ...
head signature(x = "micro_array"): ...
inference signature(M = "micro_array"): ...
plot signature(x = "micro_array", y = "ANY"): ...
plot signature(x = "micro_array", y = "ANY"): ...
plot signature(x = "micropredict", y = "ANY"): ...
predict signature(object = "micro_array"): ...
print signature(x = "micro_array"): ...
summary signature(object = "micro_array"): ...
unionMicro signature(M1 = "micro_array", M2 = "micro_array"): ...

Examples
showClass("micro_array")
Net

Simulated network data for examples.

Description

Simulated network.

Usage

data(Net)

Examples

data(Net)
str(Net)

---------

network
A network object data.

---------

Description

A network object. It is the same as the result in the vignette for the inference of the network.

Usage

data(network)

Examples

data(network)
plot(network)
print(network)
Description

2254

Objects from the Class

Objects can be created by calls of the form `new("network", ...)`.

Slots

- `network`: Object of class "matrix"
- `name`: Object of class "vector"
- `F`: Object of class "array"
- `convF`: Object of class "matrix"
- `convO`: Object of class "vector"
- `time_pt`: Object of class "vector"

Methods

- `analyze_network` signature(`Omega = "network"`): ...
- `compare` signature(`Net = "network", Net_inf = "network", nv = "Numeric"`): ...
- `cutoff` signature(`Omega = "network"`): ...
- `evolution` signature(`net = "network"`): ...
- `geneNeighborhood` signature(`net = "network"`): ...
- `plot` signature(`x = "network", y = "ANY"`): ...
- `plot` signature(`x = "network", y = "micro_array"`): ...
- `position` signature(`net = "network"`): ...
- `print` signature(`x = "network"`): ...

Examples

`showClass("network")`
network_random

Generates a network.

Description
Generates a network.

Usage

```r
network_random(nb, time_label, exp, init, regul, min_expr, max_expr, casc.level)
```

Arguments

- `nb`: Integer. The number of genes.
- `time_label`: Vector. The time points measurements.
- `exp`: The exponential parameter, as in the barabasi.game function in igraph package.
- `init`: The attractiveness of the vertices with no adjacent edges. See barabasi.game function.
- `regul`: A vector mapping each gene with its number of regulators.
- `min_expr`: Minimum of strength of a non-zero link
- `max_expr`: Maximum of strength of a non-zero link
- `casc.level`: ...

Value
A network object.

Author(s)
Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.

References


Examples

```
set.seed(1)
Net<-network_random(
  nb=100,
  time_label=rep(1:4,each=25),
  exp=1,
  init=1,
  regul=round(rexp(100,1))+1,
  min_expr=0.1,
  max_expr=2,
  casc.level=0.4
)
plot(Net)
```

Net_inf

*Reverse-engineered network of the simulated data.*

Description

The reverse-engineered network of the simulated data (M and Net).

Usage

```
data(Net_inf)
```

Examples

```
data(Net_inf)
str(Net_inf)
```

plot-methods

*Plot*

Description

Considering the class of the argument which is passed to plot, the graphical output differs.

Methods

signature(x = "micro_array", y = "ANY", ...) x a micro\_array object

- list_nv a vector of cutoff at which the network should be shown

signature(x = "network", y = "ANY", ...) x a network object

... Optionnal arguments:

- gr a vector giving the group of each gene

- choice what graphic should be plotted: either "F" (for a representation of the matrices F) or "network".
nv  the level of cutoff. Default to 0.
ini  using the “position” function, you can fix the position of the nodes
color.vertex  a vector defining the color of the vertex
ani  vector giving the size of the plot. Default to c(2000,1000)
video  if ani is TRUE and video is TRUE, the animation result is a GIF video
label_v  vector defining the vertex labels
legend.position  position of the legend
frame.color  color of the frames
label.hub  logical; if TRUE only the hubs are labeled
edge.arrow.size  size of the arrows; default to 0.7
edge.thickness  edge thickness; default to 1.
signature(x = "micropredict", y = "ANY", ...) x a micropredict object
...  Optionnal arguments: see plot for network

Examples

data(Net)
plot(Net)

data(M)
plot(M)

data(Selection)
data(network)
nv<0.11
plot(network, choice="network", gr=Selection@group, nv=nv, label_v=Selection@name,
edge.arrow.size=0.9, edge.thickness=1.5)

position-methods  Returns the position of edges in the network

Description

Returns the position of edges in the network

Methods

signature(net = "network") Returns a matrix with the position of the node. This matrix can
then be used as an argument in the plot function.

Examples

data(Net)
position(Net)
Prediction of the gene expressions after a knock-out experience

**Description**

Prediction of the gene expressions after a knock-out experience

**Usage**

```r
predict(object,...)
```

**Arguments**

- `object` : a micro_array object
- `...` : Other arguments:
  - `Omega` : a network object.
  - `nv` : numeric; the level of the cutoff
  - `targets` : [NULL] vector; which genes are knocked out?

**Author(s)**

Nicolas Jung, Frédéric Bertrand, Myriam Maumy-Bertrand.

**References**


**Examples**

```r
data(Selection)
data(network)
# A nv value can be chosen using the cutoff function
nv=.11
EGR1<-which(match(Selection@name,"EGR1") == 1)
P<-position(network,nv=nv)

# We predict gene expression modulations within the network if EGR1 is experimentally knocked-out.
prediction_ko5<-predict(Selection,network,nv=nv,targets=EGR1)

# Then we plot the results. Here for example we see changes at time point t2:
plot(prediction_ko5,time=2,ini=P,label_v=Selection@name)
```
**print-methods**

Description

Methods for function print ~~

Examples

data(Net)
print(Net)

data(M)
print(M)

**Selection**

Selection of genes.

Description

20 (at most) genes with differential expression at t1, 20 (at most) genes with differential expression at t2, 20 (at most) genes with differential expression at t3, 20 (at most) genes with differential expression at t4 et 20 (at most) genes with global differential expression were selected.

Usage

data(Selection)

Examples

data(Selection)
head(Selection)
summary(Selection,3)

**summary-methods**

Methods for Function summary

Description

Methods for function summary

Examples

data(M)
summary(M)
unionMicro-methods

Makes the union between two micro_array objects.

Description

Makes the union between two micro_array objects.

Methods

signature(M1 = "micro_array", M2 = "micro_array") Returns a micro_array object which is the union of M1 and M2.
signature(M1 = "list", M2 = "ANY") Returns a micro_array object which is the union of the elements of M1.

Examples

data(M)
#Create another microarray object with 100 genes
Mbis<-M
#Rename the 100 genes
Mbis@name<-paste(M@name,"bis")
rownames(Mbis@microarray) <- Mbis@name
#Union (merge without duplicated names) of the two microarrays.
str(unionMicro(M,Mbis))
Index

*Topic **classes**
  micro_array-class, 16  
  micropredict-class, 15  
  network-class, 18

*Topic **datasets**
  M, 15  
  Net, 17  
  Net_inf, 20  
  network, 17  
  Selection, 23

*Topic **methods**
  analyze_network, 3  
  cutoff, 6  
  dim, 7  
  evolution, 7  
  geneNeighborhood, 8  
  genePeakSelection, 9  
  head, 13  
  inference, 14  
  plot-methods, 20  
  position-methods, 21  
  predict, 22  
  print-methods, 23  
  summary-methods, 23  
  unionMicro-methods, 24

*Topic **package**
  Cascade-package, 2

analyze_network, 3  
analyze_network, network-method  
  (analyze_network), 3  
analyze_network-methods  
  (analyze_network), 3  
as.micro_array, 4  

Cascade (Cascade-package), 2  
Cascade-package, 2  
compare (compare-methods), 5  
compare, network, network, numeric-method  
  (compare-methods), 5  
compare-methods, 5  
cutoff, 6  
cutoff, network-method (cutoff), 6  
cutoff-methods (cutoff), 6  
dim, 7  
dim, micro_array-method (dim), 7  
dim-methods (dim), 7  
evolution, 7  
evolution, network-method (evolution), 7  
evolution-methods (evolution), 7  
gene_expr_simulation, 12  
gene_expr_simulation, network-method  
  (gene_expr_simulation), 12  
gene_expr_simulation-methods  
  (gene_expr_simulation), 12  
geneNeighborhood, 8  
geneNeighborhood, network-method  
  (geneNeighborhood), 8  
geneNeighborhood-methods  
  (geneNeighborhood), 8  
genePeakSelection, 9  
genePeakSelection, micro_array, numeric-method  
  (genePeakSelection), 9  
genePeakSelection-methods  
  (genePeakSelection), 9  
geneSelection (genePeakSelection), 9  
geneSelection, list, list, numeric-method  
  (genePeakSelection), 9  
geneSelection, micro_array, micro_array, numeric-method  
  (genePeakSelection), 9  
geneSelection-methods  
  (genePeakSelection), 9

head, 13  
head, ANY-method (head), 13  
head, micro_array-method (head), 13  
head-methods (head), 13
INDEX

inference, 14
inference,micro_array-method
  (inference), 14
inference-methods (inference), 14

M, 15
methods (head), 13
micro_array-class, 16
micropredict-class, 15

Net, 17
Net_inf, 20
network, 17
network-class, 18
network_random, 19

plot,ANY,ANY-method (plot-methods), 20
plot,micro_array,ANY-method
  (plot-methods), 20
plot,micropredict,ANY-method
  (plot-methods), 20
plot,network,ANY-method (plot-methods),
  20
plot-methods, 20
position (position-methods), 21
position,network-method
  (position-methods), 21
position-methods, 21
predict, 22
predict,ANY-method (predict), 22
predict,micro_array-method (predict), 22
predict-methods (predict), 22
print,ANY-method (print-methods), 23
print,micro_array-method
  (print-methods), 23
print,network-method (print-methods), 23
print-methods, 23

Selection, 23
summary,ANY-method (summary-methods), 23
summary,micro_array-method
  (summary-methods), 23
summary-methods, 23

unionMicro (unionMicro-methods), 24
unionMicro,list,ANY-method
  (unionMicro-methods), 24
unionMicro,micro_array,micro_array-method
  (unionMicro-methods), 24
unionMicro-methods, 24