exonmap

April 20, 2011

`array.subset`	*Given an expression object get a particular subset of arrays defined by the phenoData*

Description

Looks up the column named 'group' in the phenoData object to find members with a particular name and returns the specified arrays

Usage

```
array.subset(x,group,members)
```

Arguments

`x`	expression data
`group`	the column to select on
`members`	vector of column entries to pick

Value

An object with only the selected arrays in it

Author(s)

Crispin Miller

References

http://bioinformatics.picr.man.ac.uk/

See Also

`group.indices`

Examples

```
  ## Not run:
    #add data
    array.subset(exprs,"group",c("a","b"))

## End(Not run)
```

`db.local.info` *Display the contents/clear the contents of the local data directory*

Description

The `filtering` functions will make a local copy of the filtering data they need in a directory, 'db.local' in the R_XMAP_CONF_DIR, if 'db.local' exists.

`db.local.info` lists the files that have been created, and `clear.db.local.info` deletes them all. These files are relatively small, and these functions should probably never be needed.

Usage

```
db.local.info()
clear.db.local()
```

Value

Nothing.

Author(s)

C.J. Miller

References

http://xmap.picr.man.ac.uk

Examples

```
if(interactive()) {
   db.local.info()
   clear.db.local()
}
```

`details` *Get detailed annotation for exons, transcripts and genes*

Description

Given a character vector of Ensembl database names, get more detailed annotation from X:Map.

Usage

```
gene.details(v)
transcript.details(v)
exon.details(v)
```

Arguments

`v`	A character vector of database identifiers

Details

Connects to the X:Map database to retrieve data. Before these functions can be used, `xmapConnect` must have been called.

Value

A `data.frame`.

Author(s)

C.J. Miller, M.J. Okoniewski

References

`http://xmap.picr.man.ac.uk`

See Also

xmapConnect filters mappings

Examples

```
if(interactive()) {
  xmapConnect()
  gene        <-  probeset.to.gene(c("3743919"))
  transcripts <- gene.to.transcript(gene)
  exons       <- gene.to.exon(gene)
  gene.details(gene)
  transcript.details(transcripts)
  exon.details(exons)
}
```

`filters` *Filter vectors of Affymetrix Exon array probeset names according to where they target*

Description

Given a character vector of probeset names, filter it to keep (or exclude) those probesets that are mapped by X:Map to exons, introns, intergenic regions or are annotated as containing potentially cross-hybridizing (multitarget) probes. Functions of the form `is. ...` return a logical vector.

Usage

```
exonic(v,exclude=FALSE,mt.rm=TRUE)
intronic(v,exclude=FALSE,mt.rm=TRUE)
intergenic(v,exclude=FALSE,mt.rm=TRUE)
multitarget(v,exclude=FALSE,mt.level=1)
is.exonic(v)
is.intronic(v)
is.intergenic(v)
is.multitarget(v,mt.level=1)
select.probewise(v,filter=c("exonic","intronic","intergenic","multitarget"),mt
exclude.probewise(v,filter=c("exonic","intronic","intergenic","multitarget"),m
```

Arguments

`v`	A character vector of probeset names identifiers
`exclude`	If TRUE, remove (rather than keep) matching probesets, from the list.
`mt.rm`	If TRUE, remove multitarget probesets before returning the result
`mt.level`	The amount of multitargeting needed before a probeset is removed. See the package vignette for more details on how this number is calculated.
`filter`	What sort of probeset should be retained/removed?

Details

Connects to the X:Map database to retrieve data. Before these functions can be used, `xmapConnect` must have been called.

By default, multitarget probesets are removed.

Value

A character vector of filtered names.

Author(s)

C.J. Miller, M.J. Okoniewski

References

http://xmap.picr.man.ac.uk

See Also

xmapConnect mappings details

Examples

```
if(interactive()) {
  xmapConnect()
  gene <-  probeset.to.gene(c("3743919"))
  ps   <- gene.to.probeset(gene,as.vector=TRUE)
  exonic(ps)
  intronic(ps)
  intergenic(ps)
  multitarget(ps)
  exonic(ps,exclude=TRUE)
  intronic(ps,exclude=TRUE)
  intergenic(ps,exclude=TRUE)
  multitarget(ps,exclude=TRUE,mt.level=1)
  #or
  select.probewise(ps,"exonic") #etc..
  exclude.probewise(ps,"exonic") #etc..
  is.exonic(ps)
  is.intronic(ps)
  is.intergenic(ps)
  is.multitarget(ps)
  is.multitarget(ps,mt.level=4)

}
```

`gene.graph` *Use the X:Map database to find annotated gene structure and generate a plot*

Description

Draws a variety of line graphs mapping expression data to a given gene.

Usage

```
gene.graph(gene,data,gps,group,gp.cols,gp.lty,gp.pch,scale.to.gene = FALSE,type=
```

Arguments

`gene`	The gene to plot
`data`	matrix or `ExpressionSet` object containg expression data
`gps`	Either a `list` of groups by which to collect the expression data when calculating, for example, fold change or mean intensities, or, if `group` is specified, the names of items in one of the columns in `pData(x)`. See details.
`group`	If specified, then the column in `pData(x)` to use when defining the groups of arrays to compare. See details.
`gp.cols`	Vector of colours to colour each group's line by. If generating a fold or splicing index plot, only the first element is used.

gp.lty	Vector of line types for each group's line. If generating a fold change or splicing index plot, only the first element is used.
gp.pch	For 'by.order' plots, a vector of plot character types for each group's line. If generating a fold change or splicing index plot, only the first element is used.
scale.to.gene	If TRUE, then mean-center each plot around zero.
type	The type of calculatin used to create the data for the plot. See details.
use.symbol	If TRUE then label by the gene symbol, if FALSE, the gene name.
use.mt	If TRUE then include multitarget probesets. See select.probewise and exclude.probewise for details on how the filtering is done.
probes.min	Show probesets with at least this many probes hitting the gene.
main	Plot title.
xlab	X axis label. Overrides use.symbol.
ylab	Y axis label.
xlim	Range of values to plot on the x axis.
ylim	Range of values to plot on the y axis.
exon.y	y position to draw exons
exon.height	Height to draw exons.
by.order	If TRUE then the x axis position corresponds to the nucleotide position of the probeset match against the genome (see details), including introns. If FALSE, then sort probesets by chromosomal location, and plot them in numerical order.
show.introns	Only has an effect when by.order is TRUE. If FALSE then don't include intronic probesets in the plot.
exon.bg.col	Background colour used to draw exons in by.order plots. Setting the colour to NA suppresses them.
exon.bg.border.col	Border colour used to draw exons. Setting the colour to NA suppresses them.

Details

At its simplest, takes an Ensembl gene id and plots the intron-exon structure of the gene along with one or more line plots calculated from the expression data. The method used to calculate the plotted data is specified by type, and can be used to define plots based on average intensities, fold changes, or the splicing index.

The function divides the expression data into one or more groups, defined by the parameter gps.

Groups of arrays can be specified in two ways, depending on whether groups is supplied. If it is, then it should represent the name of a column in the ExpressionSet's pData object, and gps should be a list of levels in this factor defining the groups of arrays. So for example, ...,group="group",gps=c("a","b"),... will define two groups of arrays, one for each cell line, as defined by the "group" column in the expression set's pData object.

Alternatively, if groups is not supplied, gps should be a list of numeric vectors, each defining the indices of a set of arrays. For example, ...,gps=list(a=1:3,b=4:6),... would define two groups, called "a" and "b", each with three arrays in it, while ...,gps=list(1,2,3,4,5,6),... would define 6 groups, and would therefore result in a separate line for each of the individual arrays.

When the type of the plot is 'mean-int' or 'median-int' then the mean (or median) intensity for each group is plotted as a separate line in the plot. If the type is 'mean-fc', 'median-fc' or 'splicing-index', then gps is expected to contain two elements and a single line is plotted, representing the average fold change.

Note that for fold change calculations the number returned is gps[1] -gps[2] i.e. if gp[1] is more highly expressed than group 2, the result is positive.

The x position of each probeset is taken to be half way between the 5'-most and 3'-most probe for that probeset. If `by.order` is TRUE, then probesets are sorted by x position and plotted in numeric order. For these (`by.order=TRUE` plots), if `show.introns` is `FALSE`, then only exon-targeting probesets are plotted.

Value

none

Author(s)

Crispin Miller

References

http://bioinformatics.picr.man.ac.uk/

See Also

`gene.strip` `plotGene` `mappings` `filters` `details`

Examples

```
## Not run:
   xmapConnect()
   data(exonmap)
   par(mfrow=c(3,2))
   gene.graph("ENSG00000141510",x.rma,gps=list(1:3,4:6),type="mean-fc",gp.col="red")
   gene.graph("ENSG00000141510",x.rma,gps=list(1:3,4:6),type="mean-int",gp.col=c("red","c
   gene.graph("ENSG00000141510",x.rma,gps=list(1,2,3,4,5,6),type="mean-int",gp.col=1:6)
   gene.graph("ENSG00000141510",x.rma,gps=list(1,2,3,4,5,6),type="mean-int",gp.col=1:6,by
   gene.graph("ENSG00000141510",x.rma,gps=list(1,2,3,4,5,6),type="mean-int",gp.col=1:6,by
   gene.graph("ENSG00000141510",x.rma,gps=list(1,2,3,4,5,6),type="mean-int",gp.col=c(rep(

## End(Not run)
```

---

gene.legend *Generate a colour bar to use as a legend*

Description

Adds a colour bar to a plot. Used by `plotGene` and `gene.strip`

Usage

```
gene.legend(x,y,w,h,col,col.range,border="#dddddd",cex)
```

Arguments

x	x location of legend
y	y location of legend
w	width of colour bar. Defaults to 10% of the plot region
h	height of colour bar. Defaults to the height of a character
col	palette used to generate colour bar
col.range	a range specifying left and right extents of colour bar
cex	character expansion
border	Border colour for each cell in the colour ar

Details

Is called by plotGene and gene.strip by default. Position of the legend can be fine tuned by calling this function directly.

Value

none

Author(s)

Crispin Miller

References

http://bioinformatics.picr.man.ac.uk/

See Also

plotGene gene.strip

Examples

```
if(interactive()) {
  xmapConnect("human")
  data(exonmap)

  plotGene("ENSG00000141510",x.rma,gps=list(1:3,4:6),type="mean-fc",show.legend=FALSE)
  gene.legend(par()$usr[1]+1000,2,col=col.rd.bl,col.range=c(-5,5),cex=0.75)
}
```

`gene.strip` *Use the X:MAP database to find annotated gene structure and generate a plot for multiple genes*

Description

Takes a list of genes and an `ExpressionSet` object or matrix and generates a plot summarising the expression data for the supplied genes.

Usage

```
gene.strip(genes,data,gps,group,col=col.rd.bl,col.range,type=c("mean-int","media
           show.introns=FALSE,f,f.extra.params, col.f =value.to.colour, scale.to
           use.mt=FALSE,no.data.col="white",probes.min=4,main,xlab,ylab,xlim,sho

value.to.colour(vals,col=col.rd.bl,col.range=c(-5,5))
ps.value(x,gps,type=c("mean-int","median-int","mean-fc","median-fc","splicing-in
```

Arguments

`genes`	A character vector of Ensembl gene ids containing the genes to plot. Duplicates are removed, and genes plotted in order with the first gene being placed at the bottom of the plot
`data`	Expression data (should be a matrix or `ExpressionSet`). If a matrix is supplied `row.names` should correspond to probeset ids for the genes to be plotted. Note that if a probeset id is missing, the probeset will be silently ignored.
`gps`	List of groups by which to collect the expression data when calculating, for example, fold change or mean intensities. Each group is a vector of column indices into `data`
`group`	If specified, then the column in `pData(x)` to use
`col`	A vector containing the colours to use when colouring the plot by expression data. `col.rd.bl` is used by default.
`col.range`	A range specifying the extents of the colour palette. Expression data are turned into a value for each probeset (how this is done is defined by `type`) and then mapped into the colour vector `col`. `col.range` specifies the value corresponding to the first and last entry in the colour palette; values outside this range are mapped to the extreme. By default the ranges are `c(-5,5)` for fold change plots and `c(0,16)` for intensity.
`type`	The way to map the expression data onto colours. For example, `mean-fc` expects two groups and calculates the mean fold-changes between them.
`show.introns`	If `FALSE`, then draw exons in number order, all the same size. If `TRUE`, then x-position corresponds to residue position from the start of the gene, and intronic probes are also shown.
`f`	The function used to map between the expression data and a colour in `col`. By default, this is `ps.value`.
`f.extra.params`	Any extra parameters that need to be passed through to `f`. This is only necessary if supplying an alternative function for computing the colourings.

col.f	Function used to map the expression summary data generated by f to a colour in col. Not normally required; might be used for a non-linear scale, for example.
scale.to.gene	If TRUE,then mean-center the values for each gene around zero.
use.symbols	If TRUE then label the plot using gene symbols. otherwise, use the Ensembl gene id.
use.mt	If TRUE then use data from multiply targeted probesets when generating the plots. See select.probewise and exclude.probewise for more details.
no.data.col	The colour to draw exons when there is no matching probeset
probes.min	Ignore probesets unless they have at least this many probes hitting an exon or gene
main	plot title
xlab	x axis label
ylab	y axis label
vals	A numeric vector containing values that should be mapped into the specified palette
x	A vector of expression data for a probeset
xlim	range specifying x-axis limits within which to plot
show.legend	If TRUE, then plot a colour bar in the margin of the plot, showing col and the col.range

Details

At its simplest, takes a list of genes and an ExpressionSet object and plots their data coloured by expression. Each row of the plot corresponds to a gene, and the X axis corresponds to position within that gene. By default the plot shows only exons. Each exon is represented by a rectangle, coloured using the expression data (see below), and introns are ignored. Overlapping exons are plotted next to each other, and if multiple probesets hit an exon they are stacked vertically within the exon. Data are filtered (by default) to remove multiply targeted probesets. If there are no 'well-behaved' probesets hitting an exon, it is drawn as a rectangle in the 'no.data.col', which is, by default, white.

If plot.introns is TRUE then introns are included in the plot, and position on the x-axis corresponds to nucleotide position relative to the start of the gene. Exons are drawn as rectangles in the border colour - the default is black. Note that they will show up as vertical lines if the gene is long and the the graph is not very wide. Each probe is represented by a line, is coloured by expression.

Groups of arrays can be specified in two ways, depending on whether groups is supplied. If it is, then it should represent the name of a column in the ExpressionSet's pData object, and gps should be a list of levels in this factor defining the groups of arrays. So for example, ...,group="group",gps=c("a","b"),... will define two groups of arrays, one for each cell line, as defined by the "group" column in the expression set's pData object.

Alternatively, if groups is not supplied, gps should be a list of numeric vectors, each defining the indices of a set of arrays. For example, ...,gps=list(a=1:3,b=4:6),... would define two groups, called "a" and "b", each with three arrays in it.

Note that for fold change calculations the number returned is gps[1] -gps[2] i.e. if gp[1] is more highly expressed than group 2, the result is positive. With default colouring, positive values are blue, negative, red.

Colouring can be changed by supplying an alternate palette to the default (`col.rd.bl`), and alternate mappings between values and colours can be generated by supplying a different function via `col.f`. See `value.to.colour` for more details.

Value

none

Author(s)

Crispin Miller

References

http://bioinformatics.picr.man.ac.uk/

See Also

`gene.legend` `plotGene` `gene.graph`

Examples

```
if(interactive()) {
  data(exonmap)
  xmapConnect()
  genes <- probeset.to.gene(featureNames(x.rma))
  gene.strip(genes,x.rma,list(1:3,4:6),type="mean-fc",col.range=c(-2,2))
  par(mfrow=c(2,2))
  gene.strip(genes,x.rma,list(1:3),type="mean-int",col.range=c(0,16),col=heat.colors(1
  gene.strip(genes,x.rma,list(4:6),type="mean-int",col.range=c(0,16),col=heat.colors(1
  gene.strip(genes,x.rma,list(1:3),type="mean-int",col.range=c(0,16),col=heat.colors(1
  gene.strip(genes,x.rma,list(4:6),type="mean-int",col.range=c(0,16),col=heat.colors(1

  fold.changes <- apply(exprs(x.rma)[1:10,],1,ps.value,gps=list(1:3,4:6),type="mean-fc
  value.to.colour(fold.changes)
}
```

---

group.indices — *Given an expression object get the array indices for a particular set of arrays*

Description

Looks up the column named 'group' in the phenoData object to find members with a particular name and returns the indices of those arrays

Usage

```
group.indices(x,group,members)
```

Arguments

x	expression data
group	the column to select on
members	vector of column entries to pick

Value

An object with only the selected arrays in

Author(s)

Crispin Miller

References

http://bioinformatics.picr.man.ac.uk/

See Also

array.subset

Examples

```
  ## Not run:
    #add data
    group.indices(exprs,"group",c("a","b"))

## End(Not run)
```

---

mappings *Map between probes, probesets, exons, transcripts and genes*

Description

A set of functions of the form X.to.Y. That take a character vector of database identifiers of type X, and return a set of type Y, either as a character vector, or as rows in a dataframe.

Usage

```
probeset.to.probe(v,as.vector=TRUE,unique=TRUE,mt.rm=TRUE)
probeset.to.exon(v,as.vector=TRUE,unique=TRUE,subset=c("core","est","prediction"
probeset.to.transcript(v,as.vector=TRUE,unique=TRUE,subset=c("core","est","predi
probeset.to.gene(v,as.vector=TRUE,unique=TRUE,subset=c("core","est"))
exon.to.probeset(v,as.vector=TRUE,unique=TRUE,subset=c("core","est","prediction"
exon.to.transcript(v,as.vector=TRUE,unique=TRUE,subset=c("core","est","predictio
exon.to.gene(v,as.vector=TRUE,unique=TRUE,subset=c("core","est"))
transcript.to.probeset(v,as.vector=TRUE,unique=TRUE,subset=c("core","est","predi
transcript.to.exon(v,as.vector=TRUE,unique=TRUE,subset=c("core","est","predictio
transcript.to.gene(v,as.vector=TRUE,unique=TRUE,subset=c("core","est"))
```

```
gene.to.probeset(v,as.vector=TRUE,unique=TRUE,subset=c("core","est"),probes.min=
gene.to.exon(v,as.vector=TRUE,unique=TRUE,subset=c("core","est"))
gene.to.transcript(v,as.vector=TRUE,unique=TRUE,subset=c("core","est"))
symbol.to.gene(v,as.vector=TRUE,unique=TRUE)
gene.to.exon.probeset(v,unique=TRUE,probes.min=4,mt.rm=TRUE)
gene.to.exon.probeset.expr(x,v,probes.min=4,mt.rm=TRUE)
exon.to.sequence(v,as.vector=TRUE,unique=TRUE,subset=c("core","est"))
```

Arguments

v	A character vector of database identifiers
as.vector	If TRUE, return a character vector otherwise return a data.frame
unique	If TRUE, remove duplicates from the results
mt.rm	If TRUE, remove multitarget probesets before returning the result
probes.min	Only return probesets with at least this many probes
subset	Which subset of the database to perform the mapping against? core refers to Ensembl genes, est refers to Ensembl ESTs and prediction refers to Ensembl predictions.
x	An ExpressionSet object or a matrix containing expression data. If the latter, then the rownames must specify the exon array probeset names.

Details

Connects to the X:Map database to retrieve data. Before these functions can be used, xmapConnect must have been called.

By default the results are returned as a vector, and duplicate entries are removed. Note that the function probeset.to.probe, by default, also removes multitarget probesets.

gene.to.exon.probeset generates a single data.frame with mappings between genes, exons and probesets. gene.to.exon.probeset.exprs does the same but adds the expression data for the corresponding probesets onto the beginning of the data.frame.

Mappings against the prediction subset of the database are made slightly more complicated since there are no predicted genes, only transcripts and exons. In addition, their IDs are integers not strings. The additional functions genscan.label.to.id and genescan.id.to.label should be used to provide the mapping between names of the form 'GENSCAN0000...' and the integer ids used for the transcripts.

Value

A character vector, or, if as.vector=FALSE, a data.frame.

Author(s)

C.J. Miller, M.J. Okoniewski

References

http://xmap.picr.man.ac.uk

See Also

xmapConnect filters details

Examples

```
if(interactive()) {
  xmapConnect()
  probeset.to.gene(c("3743919"))
  probeset.to.gene(c("3743919"),as.vector=FALSE)
  probeset.to.gene(c("3743919","3743919"),as.vector=FALSE,unique=TRUE)
  gene <-  probeset.to.gene(c("3743919"))
  gene.to.probeset(gene,as.vector=TRUE)
  gene.to.exon.probeset(gene)
}
```

PC-class *Class "PC" represents the result of a simple pairwise comparison between two groups of arrays*

Description

Contains two slots - one for fold changes, one for t test p-scores

Objects from the Class

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

Slots

`fc` Object of class `"numeric"` log2 fold changes

`tt` Object of class `"numeric"` t test p scores

Methods

[<- `signature(x = "PC")`: Not supported

[`signature(x = "PC")`: subset by probeset

fc `signature(object = "PC")`: fold change accessor

tt `signature(object = "PC")`: t-test p score accessor

Author(s)

Crispin J. Miller

Examples

```
## Not run:
  #add data
  r <- pc(x,"group",c("a","b"))

## End(Not run)
```

pc *Simple pariwise comparisons on exon expression data*

Description

Get fold change t test p score for each probeset between a pair of arrays.

Usage

```
pc(x,group,members)
fc(object)
tt(object)
```

Arguments

x	expression data
group	the column to select on
members	vector of length 2 of column entries to compare between
object	a pairwise comparison object of class PC as produced by pc

Details

Takes the expression data in x, use the column named 'group' in its phenotypic data and compare between the two groups of arrays defined by 'members'. fc,tt can be used to extract the fold chages and p-scores out of the resultant object.

Value

A PC object

Author(s)

Crispin Miller

References

http://bioinformatics.picr.man.ac.uk/

See Also

group.indices

Examples

```
  ## Not run:
    r <- pc(exprs,"group",c("a",b"))
    fc(r)[1:1000]
    tt(r)[1:1000]

## End(Not run)
```

plotGene *Use the X:MAP database to find annotated gene structure and generate a plot*

Description

Draws a plot of a gene's structure, possibly coloured by expression data, similar to those shown in the X:Map genome browser.

Usage

```
plotGene(x, data, gps, group, scale.to.gene = FALSE,
    type = c("mean-int", "median-int", "mean-fc", "median-fc", "splicing-index")
    use.symbol = TRUE, use.mt = FALSE,
    probes.min = 4, f = ps.value, f.extra.params,
    col = col.rd.bl, col.range, col.f = value.to.colour,
    main, xlab, ylab, xlim, ylim,
    border.col = "#aaaaaa",no.data.col = "white", text.col="black",text.bg="whit
    exon.borders,
    pad=0.1,transcript.height=0.9,show.legend=TRUE)

col.rd.bl
```

Arguments

x	the Ensembl gene id of the gene to plot
data	Expression data (should be a matrix or ExpressionSet). If present, used to colour the plot
gps	Either a list of groups by which to collect the expression data when calculating, for example, fold change or mean intensities, or, if group is specified, the names of items in one of the columns in pData(x). See details.
group	If specified, then the column in pData(x) to use when defining the groups of arrays to compare. See details.
scale.to.gene	If TRUE, then mean-center each plot around zero.
type	The type of calculatin used to create the data for the plot. See details.
use.symbol	If TRUE then label by the gene symbol, if FALSE, the gene name.
use.mt	If TRUE then include multitarget probesets. See select.probewise and exclude.probewise for details on how the filtering is done.
probes.min	The minimum number of probes within a probeset that must match to an exon before it is incorporated in the plot.
f	The function used to map between the expression data and a colour in col. By default, this is ps.value.
f.extra.params	Any extra parameters that need to be passed through to f. This is only necessary if supplying an alternative function for computing the colourings.
col	A vector containing the colours to use when colouring the plot by expression data. col.rd.bl is used by default.

col.range	A range specifying the extents of the colour palette. Expression data are turned into a value for each probeset (how this is done is defined by type) and then mapped into the colour vector col. col.range specifies the value corresponding to the first and last entry in the colour palette; values outside this range are mapped to the extremes. By default the ranges are c(-5,5) for fold change plots and c(0,16) for intensity.
col.f	Function used to map the expression summary data generated by f to a colour in col. Not normally required; might be used for a non-linear scale, for example.
main	Plot title.
xlab	X axis label. Overrides use.symbol.
ylab	Y axis label.
xlim	Range of values to plot on the x axis.
ylim	Height of y-axis. By default this is just big enough to fit the gene.
border.col	Colour to use for gene, transcript and exon edges.
no.data.col	Colour to plot exons with no matching probeset after filtering using probes.min and use.mt.
text.col	Colour to label genes and transcripts.
text.bg	Label background colour for the gene label.
exon.borders	If TRUE then draw a border around exons.
pad	Vertical space to leave between each element of the plot. Character height is adjusted to be the same as pad
transcript.height	Height of each transcript. With defaults, each gene is (transcript.height + pad) * N + 3 * pad high
show.legend	If TRUE, show a colour bar as a legend in the margin of the plot.

Details

At its simplest, takes an Ensembl gene name and plots the location and structure of the gene. If data, gp1, and gp2 are specified, then colours the gene according to the expression data. By default, this is done by calculating the mean fold change for all the well behaved exon probes (i.e. those that only hit the genome, once, in an exon in the gene of interest), mapping this value to a colour and using this to paint each exon in the gene. The same is done for transcripts and genes. Other methods of colouring are specified by type, and should be self-explanatory. See the vignette for more details. If scale.to.gene is TRUE, then fold-changes (or intensities, depending on the value of type) are calculated relative to the mean fold change for the gene. Exons for which no matching probesets are found are drawn with a black border and annotated with an 'x'.

Groups of arrays can be specified in two ways, depending on whether groups is supplied. If it is, then it should represent the name of a column in the ExpressionSet's pData object, and gps should be a list of levels in this factor defining the groups of arrays. So for example, ...,group="group",gps=c("a","b"),... will define two groups of arrays, one for each cell line, as defined by the "group" column in the expression set's pData object.

Alternatively, if groups is not supplied, gps should be a list of numeric vectors, each defining the indices of a set of arrays. For example, ...,gps=list(a=1:3,b=4:6),... would define two groups, called "a" and "b", each with three arrays in it.

Note that for fold change calculations the number returned is gps[1] -gps[2] i.e. if gp[1] is more highly expressed than group 2, the result is positive. With default colouring, positive values are blue, negative, red.

Colouring can be changed by supplying an alternate palette to the default (`col.rd.bl`), and alternate mappings between values and colours can be generated by supplying a different function via `col.f`. See `value.to.colour` for more details.

Value

none

Author(s)

Crispin Miller

References

http://bioinformatics.picr.man.ac.uk/

See Also

`gene.legend` `gene.strip` `gene.graph` `mappings` `filters` `details`

Examples

```
if(interactive()) {
 xmapConnect()
 data(exonmap)
 par(mfrow=c(3,1))
 plotGene("ENSG00000141510",x.rma,gps=list(1:3,4:6),type="mean-fc")
 plotGene("ENSG00000141510",x.rma,gps=c("a","b"),group="group",type="mean-fc")
 plotGene("ENSG00000141510",x.rma,gps=list(1:3),type="mean-int",col=heat.colors(16))
 plotGene("ENSG00000141510",x.rma,gps=list(4:6),type="mean-int",col=heat.colors(16))
}
```

`probeset.stats` *Generates summary statistics showing intron, exon and gene hits for the spefied probeset list*

Description

Each column represent the uniqueness of a probeset's hits to the genome, exons or introns. Each value in a column is 0 or a positive integer. If the value is zero then one or more probes within the probeset do not match the genome (or exons, or introns). Values of 1 correspond to probesets where each probe matches once and only once; values > 1 correspond to probesets where 1 or more probes hit multiple times. For more details see the package vignette.

Usage

```
probeset.stats(v)
```

Arguments

`v`	a probeset list

Value

A data frame with gene, exon and intron matches for each probeset.

Author(s)

Crispin. J. Miller, Michal Okoniewski

References

http://bioinformatics.picr.man.ac.uk/

See Also

select.probewise exclude.probewise

Examples

```
if(interactive()) {
  xmapConnect()
  probesets <- gene.to.probeset("ENSG00000005893");
  probeset.stats(probesets)
}
```

---

probes.in.range — *Given a set of chromosome coordinates, return the genomic features within*

Description

Return the probes,probesets,exons,transcripts or genes between the specified locations.

Usage

```
probes.in.range(start, stop, strand, chr,unique=TRUE)
probesets.in.range(start, stop, strand, chr,unique=TRUE)
exons.in.range(start, stop, strand, chr,unique=TRUE)
transcripts.in.range(start, stop, strand, chr,unique=TRUE)
genes.in.range(start, stop, strand, chr,unique=TRUE)
```

Arguments

start	Starting nucleotide position
stop	Ending nucleotide position
strand	1 is forward, -1 is reverse
chr	Chromosome
unique	If TRUE remove duplicates

Details

Connects to the X:Map database to retrieve data. Before these functions can be used, `xmapConnect` must have been called.

Value

A `character` vector, of database identifiers.

Author(s)

C.J. Miller, M.J. Okoniewski

References

`http://xmap.picr.man.ac.uk`

See Also

xmapConnect

Examples

```
if(interactive()) {
  xmapConnect()
  probes.in.range(1,1000,1,"1")
  probesets.in.range(1,1000,1,"1")
  exons.in.range(1,1000,1,"1")
  transcripts.in.range(1,1000,1,"1")
  genes.in.range(1,1000,1,"1")
}
```

---

read.exon *Read a Set of .CEL Files and Phenotypic Data representing exon arrays*

Description

Reads the specified file, which defines an AnnotatedDataFrame for a set of .CEL files. Reads the specified files into an `AffyBatch` object and then creates an `AnnotatedDataframe` object, defining the experimental factors for those chips.

Usage

```
read.exon(covdesc = "covdesc", path=".", ...)
```

Arguments

`covdesc`	A white space delimited file suitable for reading as a `data.frame`. The first column (with no column name) contains the names(or paths to) the .CEL files to read. Remaining columns (with names) represent experimental factors for each chip. these become elements of the `AnnotatedDataFrame` object.
`...`	extra functions to pass on to `ReadAffy`
`path`	The path to prefix the filenames with before calling `ReadAffy`

Value

An AffyBatch object

Author(s)

Crispin J Miller

References

http://bioinformatics.picr.man.ac.uk/

See Also

ReadAffy, AffyBatch, data.frame, AnnotatedDataFrame

Examples

```
  ## Not run:
    eset <- read.exon() # read a set of CEL files

## End(Not run)
```

---

si *Calculate the splicing index*

Description

Calculates the splicing index for the probesets in one or more genes, as defined in the Affymetrix white paper "Alternative Transcript Analysis Methods for Exon Arrays".

Usage

```
si(x, v, group, gps, median.gene=FALSE,median.probeset=FALSE,unlogged=TRUE)
```

Arguments

x	eSet containing expression data
v	Character vector of Ensembl gene names
group	If defined, the column name in the ExpressionSet's pData object in which to look for gps
gps	The two sets of arrays to compare
median.gene	Use the median instead of the mean when calculating averages across genes
median.probeset	Use the median instead of the mean when calculating averages across probesets in each replicate group
unlogged	Unlog the expression data before calculating the splicing index (and then re-log afterwards)

Details

The splicing index gives a measure of the difference in expression level for each probeset in a gene between two sets of arrays, relative to the gene-level average in each set. This is calculated only for those probesets that are defined as exon targeting and non-multitargetted (See `select.probewise` and `exclude.probewise` for more details of how this filtering is performed.

The two sets of arrays can be specified in two ways: First, by using numeric indices defining the appropriate columns in the expression data. This is done by supplying these as a list to `gps` (e.g. `gps=list(1:3,4:6)` will calculate the splicing index between arrays 1,2,3 and 4,5,6. Alternatively, the annotation in the `phenoData` object from `x` can be used (e.g. `group="treatment",gps=c("a","b")` will compare between the arrays labelled "a", and "b" in the "treatment" column of `pData(x)`).

The implementation also calculates a `p.value` and `t.statistic` for each probeset; these are returned alongside the splicing index.

By default, the splicing index is calculated using the mean across genes and samples. Specifing `median.gene=TRUE` or `median.probeset=TRUE` will use the median instead (for the gene or probeset level averages, respectively). It is calculated using the unlogged data, unless `unlogged=FALSE`. This only affects the internal calculations; values in `x` are always assumed to be logged, and the splicing index is always returned on the log2 scale.

Value

A `list`, one element for each gene. Each element contains a `data.frame`, with the results for a given gene. Each row corresponds to a probeset, and there are four columns in the `data.frame`: `"si"`,`"p.value"`,`"t.statistic"` and `"gene.av"`.

Author(s)

Crispin J Miller with contributions from Carla Moller Levet and Michal J Okoniewski

References

http://bioinformatics.picr.man.ac.uk/

Examples

```
if(interactive()) {
  xmapConnect()
  data(exonmap)
  gg <- probeset.to.gene(c("2326780","2326822" ))
  spl.idx <-  si(x, gg, "group", c("a","b"))
  spl.idx <-  si(x, gg, gps=list(1:3,4:6))
}
```

`xmapDatabase` *Connect/disconnect to an xmap database*

Description

`xmapConnect` connects to an instance of the xmap database. `xmapDisconnect` disconnects. If no parameters are specified, then the user is presented with a list of possible databases to chose from. Alternatively, the name of the database can be specified. Username and password can be specified if required; they are requested if they are needed but not specified in the function call. `xmapDatabase` provides the same functionality as `xmapConnect`; it is there for backwards compatibility.

Usage

```
xmapConnect(name,username,password)
xmapDatabase(name,username,password)
xmapDisconnect()
```

Arguments

`name`	The name of the database to connect to
`username`	The username to use
`password`	The password to specify for the connection

Details

The function looks in the file `databases.txt` for a list of possible databases and their connection details. For information about the contents of this file, see the package installation instructions.

Value

Nothing.

Author(s)

C.J. Miller, M.J. Okoniewski

References

http://xmap.picr.man.ac.uk

Examples

```
if(interactive()) {
  xmapDatabase()
  xmapDisconnect()
}
```

`xmapGene` *Open a browser window at the X:Map database, centered on the specified feature*

Description

Displays gene/transcript/exon/probeset in a web browser using the X:Map genome database.

Usage

```
xmapGene(v)
xmapTranscript(v)
xmapExon(v)
xmapProbeset(v)
```

Arguments

`v`	Database identifier

Details

The function attempts to open a new window using the default web browser, pointing at the X:Map website for the item of interest.

Author(s)

CJ Miller, MJ Okoniewski

References

http://bioinformatics.picr.man.ac.uk/

Examples

```
if(interactive()) {
  xmapConnect()
  xmapGene("ENSG00000146556")
}
```

`x.rma` *Sample exonmap dataset*

Description

ExpressionSet object that include probesets for genes TP53, SULF1, MDFI, TFF3, VNN1, APOBEC3D and PGR

Usage

```
data(exonmap)
```

Details

The ExpressionSet object is a subset (7 genes, 240 probesets) of RMA processed exon array data comparing triplicate samples from the cell lines MCF7 and MCF10A.

Value

On loading, creates the object x.rma.

Author(s)

Michal Okoniewski

Index