AnnotationTrack-class

Class "AnnotationTrack"

Description
A generic object to store annotation

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

Slots
- `chr`: Object of class "numeric"
- `strand`: Object of class "numeric"
- `regions`: Object of class "dfOrNULL"
- `dp`: Object of class "DisplayPars"

Extends
Class "gdObject", directly.

Methods
- `drawGD` signature(`gdObject = "AnnotationTrack"`):...
- `getPlotId` signature(`obj = "AnnotationTrack"`):...
- `initialize` signature(`.Object = "AnnotationTrack"`):...

Author(s)
James Bullard

Examples
`showClass("AnnotationTrack")`
BaseTrack-class

Class "BaseTrack" represents base specific data

Description

Represents specific data, e.g. how many times was every base sequenced

Objects from the Class

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

Slots

- **base**: Object of class "numeric". Is a vector of base positions
- **value**: Object of class "numeric". Is a vector of corresponding values for every base
- **strand**: Object of class "character" represents that DNA strand
- **dp**: Object of class DisplayPars to control various features of how the track is displayed.

Extends

Class "gdObject", directly.

Methods

- **show** signature(object = "BaseTrack"): ...

Author(s)

Steffen Durinck

References

http://www.stat.berkeley.edu/~steffen/

See Also

objects to See Also as gdPlot

Examples

```r
if (interactive()) {
  data("exampleData", package="GenomeGraphs")
  ga <- new("GenomeAxis")
  bt <- new("BaseTrack", base = yeastCons1[,1], value = yeastCons1[,2],
            dp = DisplayPars(color = "darkblue"))
  gdPlot(list(ga, bt))
}
```
Class "DisplayPars" is used to specify graphical parameters to gdObjects.

**Description**

The `DisplayPars` functions analogously to `par` and `gp`. Generally the class is instantiated using the `DisplayPars` function rather than directly.

**Objects from the Class**

Objects can be created by calls of the form `DisplayPars(...)` rather than calls to: `new("DisplayPars", ...)` by calling the `DisplayPars` function directly in the constructor the gdObjects are guaranteed to have the appropriate defaults.

**Slots**

- **pars**: Object of class "environment" Generally this slot is not accessed directly.

**Methods**

- **getPar** signature(obj = "DisplayPars"): gets a graphical parameter by name
- **initialize** signature(.Object = "DisplayPars"): This constructor should not be called directly.
- **setPar** signature(obj = "DisplayPars"): sets a graphical parameter - see the example below. Often it is easier to set the graphical parameter from within the gdObject.
- **show** signature(object = "DisplayPars"): prints current graphical parameters

**Warning**

The `DisplayPars` class should not be manipulated directly. The preferred method for interacting with the class can be seen in the example below.

**Author(s)**

James Bullard

**Examples**

```r
showClass("DisplayPars")
if (interactive()) {
  minbase = 10000
  maxbase = 15000
  mart <- useMart("ensembl", dataset = "scerevisiae_gene_ensembl")
  genesplus <- new("GeneRegion", start = minbase, end = maxbase, biomart = mart,
                   strand = "+", chromosome = "I", dp = DisplayPars(size = 2))
  ## plot it.
  gdPlot(list(genesplus, new("Title", title = "genes")), minbase, maxbase)
  ## to obtain a list of the current graphical parameters:
  print(genesplus@dp)
```
DisplayPars

Description
DisplayPars takes any number of named arguments which will be used by the drawGD method of the gdObject. These arguments are analogous to both par and gp of the traditional and grid graphics systems respectively. Different functions support different graphical parameters - thus it is necessary to consult the documentation of the particular gdObject to determine which DisplayPars will be used.

Usage
DisplayPars(...)

Arguments
... name value pairs

Details
It is not recommended to call new("DisplayPars", ...) directly; rather this function DisplayPars() should be called instead. If a gdObject has already been instantiated then the appropriate method for changing graphical parameters is: setPar.

Value
Returns an object of type DisplayPars, generally this will be called during a call to the new function for a particular gdObject.

Author(s)
James Bullard

Examples
minbase = 10000
maxbase = 15000
mart <- useMart("ensembl", dataset = "scerevisiae_gene_ensembl")
geneplus <- new("GeneRegion", start = minbase, end = maxbase, biomart = mart,
strand = "+", chromosome = "I", dp = DisplayPars(color = "red"))
gaxis <- new("GenomeAxis", add53 = TRUE, add35 = TRUE)
genemins <- new("GenRegion", start = minbase, end = maxbase, biomart = mart,
strand = "-", chromosome = "I", dp = DisplayPars(color = "purple", size = 2))
title <- new("Title", title = "genes in a region")
gdPlot(list(genesplus, gaxis, genesminus, title), minbase, maxbase)

ExonArray-class

Class “ExonArray” representing probe level exon array data from Affymetrix

Description

Represents probe level exon array data from Affymetrix. Makes it possible to visualize alternative splicing as measured by the Affymetrix exon array platform and relate it to known transcript isoforms annotated by Ensembl

Objects from the Class

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

Slots

- intensity: Object of class "matrix", array data matrix containing probes as the rows and samples as the columns
- probeStart: Object of class "numeric" vector with the start positions of the probes
- probeEnd: Object of class "numeric" vector with the end positions of the probes
- probeId: Object of class "character" vector containing the probeset identifiers
- nProbes: Object of class "numeric" vector defining how many probes there are for each exon/probeset
- size: Object of class "numeric" specifying the size of the ExonArray plot in the final plot
- displayProbesets: Object of class "logical" used to indicate if probe set names should be plotted or not
- probesetSize: Object of class "numeric" font size of the probeset identifiers to be plotted
- color: Object of class "character" vector of colors or one color that will be used to draw the intensity values
- mapColor: Object of class "character" specifying the color of the lines that map the probeset positions to the Ensembl gene annotation
- lwd: Object of class "numeric" vector of line widths to be used to plot the intensity data.
- lty: Object of class "character" vector of line types to be used to plot the intensity data.

Extends

Class "gdObject", directly.

Methods

- show signature(object = "ExonArray"): ...

Author(s)

Steffen Durinck
Gene-class

References

http://www.stat.berkeley.edu/~steffen/

See Also

objects to See Also as `gdPlot`

Examples

```r
if(interactive()){
  data("unrData", package="GenomeGraphs")
  library(biomaRt)
  mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")

  title = new("Title", title = "ENSG00000009307", dp = DisplayPars(color = "darkslategray"))
  exmapcol = rep("khaki", length(unrNProbes))
  exmapcol[28]="darkred"
  probeSetCol = rep("grey", length(unrNProbes))
  probeSetCol[27:28]="darkslategray"
  probeSetLwd = rep(1, length(unrNProbes))
  probeSetLwd[27:28]=3

  exon = new("ExonArray", intensity = unrData, probeStart = unrPositions[,3], probeEnd=unrPositions[,4],
             mapColor = exmapcol, probeSetColor = probeSetCol, probeSetLwd = probeSetLwd), displayProbesets=FALSE)
  exon2 = new("ExonArray", intensity = unrData, probeStart = unrPositions[,3], probeEnd=unrPositions[,4],
              mapColor = exmapcol, probeSetColor = probeSetCol, probeSetLwd = probeSetLwd, plotMap=FALSE), displayProbesets=TRUE)
  affyModel = new("GeneModel", exonStart = unrPositions[,3], exonEnd = unrPositions[,4])
  gene = new("Gene", id = "ENSG00000009307", biomart = mart)
  transcript = new("Transcript", id = "ENSG00000009307", biomart = mart)
  legend = new("Legend", legend = c("affyModel", "gene"), dp = DisplayPars(color= c("darkgreen","orange"))

  gdPlot(list(title, exonarray1 = exon2, exonarray2= exon, AffymetrixModel= affyModel, gene, legend),
          minBase = min(exon@probeStart), maxBase=max(exon@probeEnd))
}
```

**Gene-class**

Class "Gene" represents the Ensembl Gene level annotation

**Description**

Class "Gene" represents the Ensembl Gene level annotation. Upon creation of an object of this class, intron and exon boundaries are retrieved from Ensembl

**Objects from the Class**

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

**Slots**

- **id**: Object of class "character", representing a unique identifier for the gene or a vector of identifiers for genes that are located near each other (or at least on the same chromosome)
- **type**: Object of class "character", representing the type of identifier used, e.g. `hgnc_symbol`, `entrezgene` and `ensembl_gene_id`. Check the `listFilters` function of the `biomaRt` package for more identifier options
**GeneModel-class**

Object of class "numeric", specifies the size of the plot

Object of class "character", specifies the color of the exons

Object of class "Mart", contains the link to the Ensembl database and should be created using the useMart function from the biomaRt package

Object of class "data.frame", contains the output from the Ensembl query, users don’t need to give a value to this

**Methods**

- initialize signature(.Object = "Gene"): ...
- drawGD signature(.Object = "Gene"): ...
- show signature(object = "Gene"): ...

**Author(s)**

Jim Bullard and Steffen Durinck

**References**

http://www.stat.berkeley.edu/~steffen/

**See Also**

objects to See Also as gdPlot

**Examples**

```r
if(interactive()){
  mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")
  gene = new("Gene", id = "ENSG00000095203", type="ensembl_gene_id", biomart = mart)
  gdPlot(list(gene), minBase= 110974000, maxBase = 111122900)
}
```

---

**GeneModel-class**  
Class "GeneModel", represents a custom gene model

**Description**

This class represents a custom gene model defined by exon boundaries. An example of this class could be an Affymetrix gene model used to create the Affy Exon array

**Objects from the Class**

Objects can be created by calls of the form new("GeneModel", ...).
Slots

- **exonStart**: Object of class "numeric", vector containing the start positions of the exons that are to be drawn
- **exonEnd**: Object of class "numeric", vector containing the end positions of the exons that are to be drawn
- **chromosome**: Object of class "numeric", chromosome name
- **dp**: Object of class "DisplayPars", color of the exons and size of the exon model in the final plot

Methods

No methods defined with class "GeneModel" in the signature.

Author(s)

Steffen Durinck

References

http://www.stat.berkeley.edu/~steffen/

See Also

objects to See Also as gdPlot

Examples

data("unrData", package="GenomeGraphs")
affyModel = new("GeneModel", exonStart = unrPositions[,3], exonEnd = unrPositions[,4])
gdPlot(list(affyModel), minBase = min(unrPositions[,3]), maxBase=max(unrPositions[,4]))

---

GeneRegion-class

Class "GeneRegion", representing gene structures in a defined genomic region

Description

Given a start and end position and a chromosome name, all gene structures in this region will be retrieved from Ensembl upon creation of the object.

Objects from the Class

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

- **start**: Object of class "numeric", start position
- **end**: Object of class "numeric", end position
- **chromosome**: Object of class "character", chromosome name
- **strand**: Object of class "character", represents the strand from which the gene structures should be retrieved. Value is either + or -
- **size**: Object of class "numeric", represents the size of the GeneRegion in the final plot
- **color**: Object of class "character", represents the color to be used to plot the exons
- **biomart**: Object of class "Mart", containing the link to the Ensembl database. This should be created by the useMart function from the biomaRt package
- **ens**: Object of class "data.frame", output of the biomaRt query, should not be used by users

Methods

- **drawGD** signature(.Object = "GeneRegion"):
- **initialize** signature(.Object = "GeneRegion"):
- **show** signature(object = "GeneRegion"):

Author(s)

Steffen Durinck

References

http://www.stat.berkeley.edu/~steffen/

See Also

objects to See Also as gdPlot

Examples

```r
if(interactive()){  
  mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")  
  plusStrand = new("GeneRegion", chromosome = "17", start = 30450000, end = 30550000, strand = "+", biomart = mart)  
  genomeAxis = new("GenomeAxis", add53=TRUE)  
  gdPlot(list(genomeAxis, plusStrand), minBase = 30450000, maxBase = 30550000)  
}
```

---

**GenericArray-class**  
Class “GenericArray”, representing array data

Description

The Generic Array class is a class that can be used to create plots from array data such as microarrays and arrayCGH platforms. It can represent the data as line plots or dot plots and segments can be included as well.
Objects from the Class

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

Slots

- **intensity**: Object of class "matrix", matrix containing the intensities of expression or cgh data. Rows should be probes, columns samples
- **probeStart**: Object of class "numeric", start position of the probes
- **probeEnd**: Object of class "numeric", end position of the probes if available
- **type**: Object of class "character", has two values: line and point. If line is selected then a line will be plotted for each sample. If point is selected, points will be plotted.
- **segments**: Object of class "list", if segments are available they have to be represented as a list, each position in the list should correspond to each sample as defined by the columns of the intensity matrix
- **segmentStart**: Object of class "list", containing the start positions of the segments
- **segmentEnd**: Object of class "list", containing the end positions of the segments
- **color**: Object of class "character", represents the color to be used to plot the intensity matrix, can be a vector of colors for multiple samples
- **lty**: Object of class "character", if line is selected as type, lty specifies which line type should be used. Can be a vector of line types for multiple samples
- **pch**: Object of class "numeric", if point is selected as type then pch represents which symbol should be used to plot the points
- **pointSize**: Object of class "numeric", specifies the point size if point is selected as type
- **lwd**: Object of class "numeric", specifies the line width if line is selected as type. Can be a vector of line widths if multiple samples are present
- **size**: Object of class "numeric" representing the size of the Array plot in the final plot
- **segmentColor**: Object of class "character", If segments are to be plotted, this will define the color of the segment lines

Methods

- `show` signature(`object = "GenericArray"`): ...

Author(s)

Steffen Durinck

References

http://www.stat.berkeley.edu/~steffen/

See Also

objects to See Also as `gdPlot`
Examples

if(interactive()){
  data("exampleData", package="GenomeGraphs")
  minbase <- 180292097
  maxbase <- 180492096
  ideog <- new("Ideogram", chromosome = "3")
  expres <- new("GenericArray", intensity = intensity, probeStart = exonProbePos,
          dp = DisplayPars(color="darkred", type="point"))
  gdPlot(list(ideog, expres), minBase = minbase, maxBase = maxbase)
}

GenomeAxis-class  Class "GenomeAxis", representing a genomic coordinate axis

Description

Represents a genomic coordinate axis

Objects from the Class

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

Slots

- `add53`: Object of class "logical", indicating if 5 to 3 prime direction needs to be plotted
- `add35`: Object of class "logical", indicating if 3 to 5 prime direction needs to be plotted
- `dp`: Object of class "DisplayPars", containing the display parameters such as size of the plot and color
- `littleTicks`: Object of class "logical", indicating if the genome axis should be dense for improved locating of regions of interest.

Methods

No methods defined with class "GenomeAxis" in the signature.

Author(s)

Steffen Durinck

References

http://www.stat.berkeley.edu/~steffen/

See Also

objects to See Also as `gdPlot`
HighlightRegion-class

Class "HighlightRegion" is used to highlight vertical blocks of genomic regions.

Description

HighlightRegion is used to highlight a genomic region of interest. The class offers the ability to highlight or block out regions of interest.

Objects from the Class

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

Slots

- **start**: Object of class "numeric" genomic start position.
- **end**: Object of class "numeric" genomic end position.
- **region**: Object of class "numericOrNull" start and end number of the tracks to be covered by the region. These start from the first track (top = 1) to the last track: length(gdObjects) in the call to gdObject
- **coords**: Object of class "character" can be either "genomic" or "absolute", if the coordinates are "absolute" then one can plot things using the coordinate space defined by: lower-left (0,0) upper-right (1,1). In this case, start = x0, end = x1 and then region = (y0, y1). See the examples for more details.
- **dp**: Object of class "DisplayPars" specifies the various display parameters.

Extends

Class "gdObject", directly.

Methods

No methods defined with class "HighlightRegion" in the signature.

Author(s)

James Bullard
Examples

```r
if (interactive()) {
  data("exampleData", package="GenomeGraphs")

  ga <- new("GenomeAxis")
  grF <- new("GeneRegion", start = 10000, end = 20000, chromosome = "I", strand = "+", biomart = yeastMart)
  grR <- new("GeneRegion", start = 10000, end = 20000, chromosome = "I", strand = "-", biomart = yeastMart)
  bt <- new("BaseTrack", base = yeastCons1[,1], value = yeastCons1[,2])
  hr1 <- new("HighlightRegion", start = 11000, end = 13000, dp = DisplayPars(alpha = 1, color = "red", lty = "dashed", lwd = 3))
  hr2 <- new("HighlightRegion", start = 15900, end = 16500)

  gdPlot(list(grF, ga, grR, bt), highlightRegions = list(hr1, hr2))
}
```

Ideogram-class  Class "Ideogram", represent an Ideogram

Description

An ideogram is a representation of a chromosome containing the banding pattern. Note that currently ideograms are only available for hsapiens.

Objects from the Class

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

Slots

- **chromosome**: Object of class "character", representing the chromosome that needs to be drawn. E.g. 3 if chromosome 3 needs to be drawn or Y for Y chromosome.
- **dp**: Object of class "DisplayPars", can be used to specify the size (default 1) of the ideogram in the final plot and to specify the highlighting color

Methods

No methods defined with class "Ideogram" in the signature.

Author(s)

Steffen Durinck

References

http://www.stat.berkeley.edu/~steffen/

See Also

objects to See Also as `gdPlot`
Examples

if(interactive()){  
data("exampleData", package="GenomeGraphs")

minbase <- 180292097
maxbase <- 180492096
ideog <- new("Ideogram", chromosome = "3")
expres <- new("GenericArray", intensity = intensity, probeStart = exonProbePos,
             dp = DisplayPars(color="darkred", type="point"))
gdPlot(list(ideog, expres), minBase = minbase, maxBase = maxbase)
}

Legend-class

Class "Legend", represents a legend to add to a plot

Description

This class represents a legend

Objects from the Class

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

Slots

  legend: Object of class "character", vector with names of the items in the legend
  dp: Object of class "DisplayPars" size of the legend (size), the size of the font (cex) and the
       colors (color) of the legend

Methods

No methods defined with class "Legend" in the signature.

Author(s)

Steffen Durinck

References

http://www.stat.berkeley.edu/~steffen/

See Also

objects to See Also as gdPlot

Examples

showClass("Legend")
### MappedRead-class

Represents mapped reads

**Description**

Represents mapped reads

**Slots**

- **start**: Object of class "numeric", containing start position of the reads
- **end**: Object of class "numeric", containing end position of the reads
- **strand**: Object of class "numeric", containing strand to which the reads map
- **chromosome**: Object of class "numeric", containing chromosome to which the reads map

**Methods**

- `show` signature(object = "MappedRead"): ...

**Author(s)**

Steffen Durinck

**References**

http://www.stat.berkeley.edu/~steffen/

**Examples**

```r
## maybe str(MappedRead) ; plot(MappedRead) ...
```

### Overlay-class

Class "Overlay"

**Description**

Superclass of overlay objects.

**Objects from the Class**

Objects from this class are generally not created.

**Slots**

- **dp**: Object of class "DisplayPars"

**Extends**

Class "gdObject", directly.
Methods

No methods defined with class "Overlay" in the signature.

Examples

showClass("Overlay")

RectangleOverlay-class
Class "RectangleOverlay"

Description

Rectangular Overlay

Objects from the Class

Objects can be created by calls of the form makeRectangleOverlay (makeRectangleOverlay).

Slots

start: Object of class "numeric" ~
end: Object of class "numeric" ~
region: Object of class "numericOrNull" ~
coords: Object of class "character" ~
dp: Object of class "DisplayPars" ~

Extends

Class "Overlay", directly. Class "gdObject", by class "Overlay", distance 2.

Methods

drawOverlay signature(obj = "RectangleOverlay"): ...

Examples

showClass("RectangleOverlay")
Segmentable-class

Class "Segmentable"

Description

The Segmentable class is used as an interface to determine whether or not a segmentation should be plotted.

Objects from the Class

This object is mostly intended to be extended.

Slots

segmentation: Object of class "SegmentationOrNULL"

Methods

getSegmentation signature(obj = "Segmentable"): returns the segmentation, this is essentially the method which is implemented by subclasses.

See Also

Segmentation

Examples

showClass("Segmentable")

Segmentation-class

Class "Segmentation" is used to specify segmentations to any class that extends Segmentable (GenericArray, BaseTrack)

Description

A Segmentation object provides line segments to various gdObjects

Objects from the Class

Objects can be created by calls of the form new("Segmentation", segments = list(1), segmentStart = list(1000), segmentEnd = list(1010)).

Slots

segments: Object of class "list" ~
segmentStart: Object of class "list" ~
segmentEnd: Object of class "list" ~
dp: Object of class "DisplayPars" ~
TextOverlay-class

Description

Textual overlay classes

Objects from the Class

Objects can be created by calls of the form `makeTextOverlay`

Slots

- `text`: Object of class "character"
- `xpos`: Object of class "numeric"
- `ypos`: Object of class "numeric"
- `region`: Object of class "numericOrNull"
- `coords`: Object of class "character"
- `dp`: Object of class "DisplayPars"

Extends

Class "Overlay", directly. Class "gdObject", by class "Overlay", distance 2.

Methods

- `drawOverlay` signature(obj = "TextOverlay"): ...

Examples

`showClass("TextOverlay")`
Title-class

Class "Title" representing the title of a plot

Description
Represent the title of a plot

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

Slots
- `title`: Object of class "character" which will be used as title
- `dp`: Object of class "DisplayPars" specifying the size and color of the title in the final plot

Methods
No methods defined with class "Title" in the signature.

Author(s)
Steffen Durinck

References
http://www.stat.berkeley.edu/~steffen/

See Also
objects to See Also as `gdPlot`

Examples
`showClass("Title")`

Transcript-class

Represent known transcript isoforms as annotated by Ensembl

Description
Represent known transcript isoforms as annotated by Ensembl

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

id: Object of class "character", represents the gene identifier that should be used to retrieve the transcript level annotation

type: Object of class "character", represents the type of identifiers used to specify the gene, e.g. hgnc_symbol, entrezgene and ensembl_gene_id

size: Object of class "numeric", represents the size of the plot of this object

transcriptSize: Object of class "numeric", represents the size of the transcripts in the plot

numOfTranscripts: Object of class "numeric", should not be used by users

color: Object of class "character", color of the exons

biomart: Object of class "Mart", containing the links to the Ensembl database. This object should be created with the useMart function of the bioMaRt package

ens: Object of class "data.frame", should not be used by the users. Contains the output from the bioMaRt query

Methods

drawGD signature(.Object = "Transcript"): ...

initialize signature(.Object = "Transcript"): ...

show signature(object = "Transcript"): ...

Author(s)

Steffen Durinck

References

http://www.stat.berkeley.edu/~steffen/

See Also

objects to See Also as gdPlot

Examples

if(interactive()){
data("unrData", package="GenomeGraphs")
mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")
transcript = new("Transcript", id ="ENSG00000009307", biomart = mart)
gdPlot(list(transcript), minBase = min(exon@probeStart), maxBase=max(exon@probeEnd))
}
Description

Upon creation of this object, transcripts present in a specified region will be retrieved from Ensembl.

Objects from the Class

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

Slots

- `start`: Object of class "numeric", the start base of the genomic region
- `end`: Object of class "numeric", the end base of the genomic region
- `chromosome`: Object of class "character", the chromosome
- `size`: Object of class "numeric", the size of the plot for this object
- `biomart`: Object of class "Mart", contains link to Ensembl and should be created using the `useMart` function of the `biomaRt` package
- `ens`: Object of class "data.frame", users should not specify this, it contains the output of the query to Ensembl

Methods

- `show` signature(object = "TranscriptRegion"): ...

Author(s)

Steffen Durinck

References

http://www.stat.berkeley.edu/~steffen/

See Also

- objects to See Also as `gdPlot`

Examples

`showClass("TranscriptRegion")`
cn

Contains dummy copy number data

**Description**

Contains dummy copy number data

**Examples**

```
#
```

drawGD

*Generic called on each gdObject to do the plotting.*

**Description**

This generic gets called on each of the gdObjects (obviously the generic is implemented by a method for each object) and thus if a user wishes to implement new gdObjects they need to have access to this generic.

**Usage**

```
drawGD(gdObject, minBase, maxBase, vpPosition, ...)
```

**Arguments**

- `gdObject`
- `minBase`
- `maxBase`
- `vpPosition`
- `...` Ignored

exonProbePos

*Contains dummy exon probe positions*

**Description**

Contains dummy exon probe positions

**Examples**

```
#
```
Class "gdObject" is the parent class of all of the objects in the system.

Description

The gdObject is the superclass of all the classes in the system and provides some basic functionality for displaying and managing graphical parameters.

Objects from the Class

Objects can be created by calls of the form `new("gdObject", ...)`. Generally, this class is meant to be subclassed and not created directly.

Slots

- **dp**: Object of class "DisplayPars"

Methods

- `getCex` signature(obj = "gdObject")
- `getColor` signature(obj = "gdObject")
- `getLty` signature(obj = "gdObject")
- `getLwd` signature(obj = "gdObject")
- `getPar` signature(obj = "gdObject")
- `getPch` signature(obj = "gdObject")
- `getSize` signature(obj = "gdObject")
- `initialize` signature(.Object = "gdObject")
- `setPar` signature(obj = "gdObject")
- `showDisplayOptions` signature(obj = "gdObject")
- `showDisplayOptions` signature(obj = "character")

Author(s)

James Bullard

Examples

`showClass("gdObject")`
gdPlot is the main plotting function of the GenomeGraphs package

Description

gdPlot is the main plotting function of the GenomeGraphs package. A collection of objects are given as a list and these will then be plotted in the order given.

Usage

gdPlot(gdObjects, minBase = NA, maxBase = NA, overlays = NULL, labelColor = "black", labelCex = 1, labelRot = 90)

Arguments

- **gdObjects**: This is either a list of gdObjects which will be plotted from top to bottom or a single gdObject to be plotted.
- **minBase**: minBase defines the minimum base that will be plotted, if omitted a minimum is determined from the objects in gdObjects if possible.
- **maxBase**: maxBase defines the maximum base that will be plotted, if omitted a minimum is determined from the objects in gdObjects if possible.
- **overlays**: overlays defines a set of regions to overlay on the plot. This argument is either a list or a single Overlay object.
- **labelColor**: Draw the labels with the given colors.
- **labelCex**: Character expansion factor.
- **labelRot**: Rotate the track labels labelRot degrees.

Author(s)

Steffen Durinck and James Bullard

References

http://www.stat.berkeley.edu/~steffen/

Examples

data("exampleData", package="GenomeGraphs")

minbase = min(probestart)
maxbase = max(probestart)

mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")

genesplus = new("GeneRegion", start = minbase, end = maxbase, strand = "+", chromosome = 
genesmin = new("GeneRegion", start = minbase, end = maxbase, strand = "-", chromosome = 

seg <- new("Segmentation", segments = segments, 
    segmentStart = segStart, segmentEnd = segEnd, dp = DisplayPars(color = "dodgerblue2", lwd=2,lty = "dashed"))

cop <- new("GenericArray", intensity = cn, probeStart = probestart,
geneBiomart

AnnotationTrack objects from biomaRt

Description

Convenience function to construct an AnnotationTrack object from biomaRt.

Usage

geneBiomart(id, biomart, type = "ensembl_gene_id", dp = NULL)

Arguments

id ~Describe id here~~
biomart ~Describe biomart here~~
type ~Describe type here~~
dp ~Describe dp here~~

Value

An AnnotationTrack object

Author(s)

James Bullard

geneRegionBiomart

Construct an AnnotationTrack object from biomaRt.

Description

This function constructs an AnnotationTrack object from Biomart. It is a convenience function.

Usage

geneRegionBiomart(chr, start, end, strand, biomart, dp = NULL, chrFunction = function(x) x, strandFunction = function(x) x)
getPar

Retrieves a display parameter from an object.

Description

Retrieves a display parameter from an object.

Usage

getPar(obj, name, ...)

Arguments

obj A gdObject or DisplayPars object.
name Name of parameter to return.
... Ignored

Examples

a <- new("GenomeAxis")
getPar(a, "size")
ideogramTab

Contains info to plot ideograms

Description

Contains info to plot ideograms

Format

The format is: chr "ideogramTab"

Source

NCBI

Examples

data(ideogramTab)
## maybe str(ideogramTab) ; plot(ideogramTab) ...

intensity

Contains dummy intensity data

Description

Contains dummy intensity data

Examples

#

makeAnnotationTrack

Create objects of class AnnotationTrack

Description

Convenience function for constructing objects of class AnnotationTrack.

Usage

makeAnnotationTrack(regions = NULL, chr = NULL, strand = NULL, start = NULL, end = NULL)
Arguments

regions A dataframe with columns start, end, feature, group, ID. start and end delineate
the boundaries of the boxes feature can be used to color the boxes. Group de-
notes linking so generally exons from a gene form a group. Finally, ID can be
used to plot names on boxes.

chr The chromosome of the regions (can be ignored)
strand The strand of the regions (can be ignored)
start If regions is missing then we construct a dataframe from the remaining para-

ters.
end Construct regions with this vector
feature Construct regions with this feature vector or scalar

Value

Returns an object of class AnnotationTrack

Examples

a <- makeAnnotationTrack(start = c(10, 15, 25), end = c(12, 19, 31),
group = c(1,1,2), feature = c("gene", "gene", "tf"),
ID = paste("id", 1:3, sep = ""), dp = DisplayPars(gene = 'blue'))
gdPlot(a, minBase = 0, maxBase = 40)

Value

Returns an object of class AnnotationTrack

Examples

a <- makeAnnotationTrack(start = c(10, 15, 25), end = c(12, 19, 31),
group = c(1,1,2), feature = c("gene", "gene", "tf"),
ID = paste("id", 1:3, sep = ""), dp = DisplayPars(gene = 'blue'))
gdPlot(a, minBase = 0, maxBase = 40)

makeBaseTrack Creates an object of class BaseTrack

Description

Creates an object of class BaseTrack, which can represent many datasets containing a base given
by a vector of positions and a corresponding vector with values for these base positions

Usage

makeBaseTrack(base, value, strand, segmentation, dp = NULL)

Arguments

base Numeric vector of base positions
value Numeric vector with values for these base positions
strand Character either + or - representing the strand
segmentation Object of class Segmentation, used when segments are needed to be drawn
dp Object of class DisplayPars representing the display parameters of the plot
makeExonArray

Value

Object of class BaseTrack

Author(s)

Jim Bullard and Steffen Durinck

References

~put references to the literature/web site here ~

See Also

DisplayPars, gdPlot

Examples

```r
##---- 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 (base, value, strand, segmentation, dp = NULL)
##{
##   pt <- getClass("BaseTrack")@prototype
##   if (is.null(dp))
##     dp <- pt@dp
##   if (missing(strand))
##     strand <- pt@strand
##   if (missing(segmentation))
##     segmentation <- pt@segmentation
##   if (missing(base))
##     stop("Need base argument to know the base positions to plot the data on the genome")
##   if (missing(value))
##     stop("Need value argument")
##   new("BaseTrack", base = base, value = value, strand = strand,
##       dp = dp, segmentation = segmentation)
## }
```

makeExonArray  Creates and object of class ExonArray

Description

Creates an object of class ExonArray, representing exon array microarray data

Usage

```r
makeExonArray(intensity, probeStart, probeEnd, probeId, nProbes, displayProbesets, 
```
Arguments

intensity  Matrix of intensities, probes in the rows, samples in the columns
probeStart Vector of probe start positions
probeEnd  Vector of probe end positions (optional)
probeId  Character vector containing the probe identifiers
nProbes  Vector indicating how many probes are in each probeset
displayProbesets  Logical indicating if the probeset identifier should be displayed or not
dp  Object of class DisplayPars to set the display parameters

Value

Object of ExonArray class

Author(s)

Steffen Durinck and Jim Bullard

References

~put references to the literature/web site here ~

See Also

gdPlot

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 (intensity, probeStart, probeEnd, probeId, nProbes,  
displayProbesets = FALSE, dp = NULL)  
{
  pt <- getClass("ExonArray")@prototype
  if (is.null(dp))
    dp <- pt@dp
  if (missing(probeEnd))
    probeEnd <- pt@probeEnd
  if (missing(probeId))
    probeId <- pt@probeId
  if (missing(nProbes))
    nProbes <- pt@nProbes
  if (is.null(dp))
    dp <- getClass("ExonArray")@prototype@dp
  new("ExonArray", intensity = intensity, probeStart = probeStart,  
      probeEnd = probeEnd, probeId = probeId, nProbes = nProbes,  
      displayProbesets = displayProbesets, dp = dp)
  
}
**makeGene**

*Creates an object of class Gene*

**Description**

Creates an object of class Gene. This represents a gene structure as annotated in Ensembl.

**Usage**

```r
makeGene(id, type, biomart, dp = NULL)
```

**Arguments**

- **id**: An identifier used to specify of which gene the intron-exon structure should be retrieved.
- **type**: The type of identifiers used, examples are ensembl_gene_id, hgnc_symbol, entrezgene. See listAttributes function of the biomaRt package for more info.
- **biomart**: Mart object, created by the useMart function of biomaRt.
- **dp**: object of class DisplayPars, determines the display of features on the plot.

**Value**

An object of class Gene

**Author(s)**

Steffen Durinck and Jim Bullard

**References**

~put references to the literature/web site here ~

**See Also**

`gdPlot`

**Examples**

```r
###---- 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 (id, type, biomart, dp = NULL)
{
  if (missing(id))
    stop("Need to specify a gene identifier for creating a Gene")
  pt <- getClass("Gene")@prototype
  if (is.null(dp))
    dp <- pt@dp
  if (missing(type))
    type = pt@type
```
makeGeneModel

new("Gene", id = id, type = type, biomart = biomart, dp = dp)
}

makeGeneModel  

Creates an object of class GeneModel

Description

Creates an object of class GeneModel representing a custom annotation or gene model

Usage

makeGeneModel(start, end, chromosome, dp = NULL)

Arguments

start  
Vector of start positions for exons

dp  
Display parameters represented as an object of class DisplayPars

dp  
Vector of end positions for exons

chromosome  
chromosome name

Value

Object of class GeneModel

Author(s)

Steffen Durinck and Jim Bullard

References

~put references to the literature/web site here ~

See Also

DisplayPars

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 (start, end, chromosome, dp = NULL)
{
  if (is.null(dp))
    dp <- getClass("GeneModel")@prototype@dp
  new("GeneModel", exonStart = start, exonEnd = end, dp = dp)
}
makeGeneRegion

Creates an object of class Gene containing the intron-exon structures of genes

Description

Creates an object of class Gene containing the intron-exon structures of genes. Given a start and end position, strand and chromosome, all the intron-exon structures of all genes laying in this region will be retrieved.

Usage

```r
makeGeneRegion(start, end, chromosome, strand, biomart, dp = NULL)
```

Arguments

- `start`: Start position on chromosome
- `end`: End position on chromosome
- `chromosome`: Chromosome name
- `strand`: Strand either + or -
- `biomart`: Mart object, created by the useMart function of biomaRt
- `dp`: Object of class DisplayPars, determines the display of features on the plot

Value

An object of class Gene

Author(s)

Steffen Durinck and Jim Bullard

References

~put references to the literature/web site here ~

See Also

- `gdPlot`

Examples

```r
## 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 (start, end, chromosome, strand, biomart, dp = NULL)
{
  if (missing(start))
    stop("Need to specify a start for creating a GeneRegion")
  pt <- getClass("GeneRegion")@prototype
```
if (is.null(dp))
  dp <- pt@dp
if (is.numeric(chromosome))
  chromosome <- as.character(chromosome)
new("GeneRegion", start = start, end = end, chromosome = chromosome,
    strand = strand, biomart = biomart, dp = dp)

makeGenericArray  Creates an object of class GenericArray

Description

Creates an object of class Generic Array representing microarray data. This could be gene expression, array CGH, etc.

Usage

makeGenericArray(intensity, probeStart, probeEnd, segmentation, dp = NULL)

Arguments

  intensity  Matrix of intensities, probes in the rows, samples in the columns
  probeStart Vector of start positions for the probes
  probeEnd   Vector of end positions for probes (optional)
  segmentation Object of class Segmentation, needs to be added if segments should be plotted as well
  dp         Object of class DisplayPars which handles the display parameters for plotting

Value

Object of class GenericArray

Warning

....

Note

~~further notes~~

~Make other sections like Warning with

0.1  Warning

.... ~

Author(s)

Jim Bullard and Steffen Durinck
makeGenomeAxis

References
~put references to the literature/web site here~

See Also
~~objects to See Also as 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 (intensity, probeStart, probeEnd, segmentation, dp = NULL)
{
  pt <- getClass("GenericArray")@prototype
  if (is.null(dp))
    dp <- pt@dp
  if (missing(probeEnd))
    probeEnd <- pt@probeEnd
  if (missing(segmentation))
    segmentation <- pt@segmentation
  if (missing(probeStart))
    stop("Need probeStart argument to know where to plot the data on the genome")
  new("GenericArray", intensity = intensity, probeStart = probeStart,
  probeEnd = probeEnd, dp = dp, segmentation = segmentation)
}

makeGenomeAxis  Creates an object of class GenomeAxis

Description
Creates an object of class GenomeAxis, representing a genome coordinate axis.

Usage
makeGenomeAxis(add53 = FALSE, add35 = FALSE, littleTicks = FALSE, dp = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add53</td>
<td>Add a 5 to 3 prime label</td>
</tr>
<tr>
<td>add35</td>
<td>Add a 3 to 5 prime label</td>
</tr>
<tr>
<td>littleTicks</td>
<td>Add smaller ticks between larger ticks</td>
</tr>
<tr>
<td>dp</td>
<td>Set the display parameters see DisplayPars</td>
</tr>
</tbody>
</table>

Value
Object of class GenomeAxis
Author(s)

Jim Bullard and Steffen Durinck

References

~put references to the literature/web site here ~

See Also

DisplayPars, gdPlot

Examples

```r
## 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 (add53 = FALSE, add35 = FALSE, littleTicks = FALSE, 
  dp = NULL)
{
  if (is.null(dp))
    dp <- getClass("GenomeAxis")@prototype@dp
  new("GenomeAxis", add53 = add53, add35 = add35, dp = dp)
}
```

`makeIdeogram` *Creates object of class Ideogram*

Description

Creates object of class Ideogram

Usage

`makeIdeogram(chromosome, dp = NULL)`

Arguments

- `chromosome`: Chromosome to represent (currently human only)
- `dp`: Display parameters such as color and size

Value

Object of class Ideogram

Author(s)

Jim Bullard and Steffen Durinck

References

~put references to the literature/web site here ~
makeLegend

See Also
gdPlot

Examples

```r
##---- 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 (chromosome, dp = NULL)
{
  if (missing(chromosome))
    stop("Need to specify chromosome for creating an Ideogram")
  if (is.numeric(chromosome)) {
    chromosome = as.character(chromosome)
  }
  if (is.null(dp))
    dp <- getClass("Iedogram")@prototype@dp
  new("Iedogram", chromosome = chromosome, dp = dp)
}
```

makeLegend  

Creates an object of class Legend

Description

Creates an object of class Legend which can be used to plot a legend

Usage

makeLegend(text, fill, cex)

Arguments

text  Vector of characters representing the legend
fill Vector of colors to fill the legend boxes
cex  Font size of the legend

Value

Object of class Legend

Author(s)

Jim Bullard and Steffen Durinck

References

~put references to the literature/web site here ~
makeRectangleOverlay

Create a rectangular overlay

Description

Construct rectangular overlays.

Usage

makeRectangleOverlay(start, end, region = NULL, coords = c("genomic", "absolute")

Arguments

- **start**: Start position in coords coordinates
- **end**: End position in coords coordinates
- **region**: Which tracks to span, or the y (vertical range)
- **coords**: Which coordinate system to use, if absolute then the range is from 0,1 and region become the y coordinates
- **dp**: The display parameters

Details

The rectangular overlay can be used to plot overlays in either genomic or absolute coordinates. If coordinates are absolute then the region argument becomes the y arguments.

Value

An object of class RectangleOverlay
**Examples**

data("exampleData", package = "GenomeGraphs")
cop <- makeGenericArray(intensity = cn, probeStart = probestart,
                        dp = DisplayPars(size=3, color = "seagreen", type="dot"))
gdPlot(list(makeGenomeAxis(), cop), overlays =
        makeRectangleOverlay(start = 180350000, end = 180350000 + 1e5, dp = DisplayPars(alpha = 0.3)))

**Description**

Construct objects of class segmentation

**Usage**

makeSegmentation(start, end, value, dp = NULL)

**Arguments**

- **start**: Either a list or a vector. If it is a list then it is a list of vectors of start position (this is the way it is represented in the segmentation class) If it is a vector it is a vector of start positions.
- **end**: Same as start, but the corresponding end positions.
- **value**: The y value of the segmentation, ie. segments(start[i], value[i], end[i], value[i])
- **dp**: The Display parameters.

**Value**

An object of class Segmentation

**Examples**

data("exampleData", package="GenomeGraphs")
seg <- makeSegmentation(segStart, segEnd, segments,
                        dp = DisplayPars(color = "black", lwd=2, lty = "solid"))
cop <- makeGenericArray(intensity = cn, probeStart = probestart,
                        segmentation = seg, dp = DisplayPars(size=3, color = "seagreen"),}
gdPlot(cop)
makeTextOverlay  
Create objects of class TextOverlay

**Description**
Create objects of class TextOverlay

**Usage**

```r
makeTextOverlay(text, xpos, ypos, region = NULL, coords = c("genomic", "absolute"), dp = NULL)
```

**Arguments**
- `text`: The text to plot
- `xpos`: The xposition of the text
- `ypos`: The yposition of the text
- `region`: The display parameters
- `coords`: The display parameters
- `dp`: The display parameters

**Value**
Returns class of TextOverlay

**Examples**
```r
data("exampleData", package="GenomeGraphs")
seg <- makeSegmentation(segStart, segEnd, segments,
    dp = DisplayPars(color = "black", lwd=2,lty = "solid"))
cop <- makeGenericArray(intensity = cn, probeStart = probestart,
    segmentation = seg, dp = DisplayPars(size=3, color = "seagreen"),
    gdPlot(cop, overlay = makeTextOverlay("Overlay Text", xpos = .5, ypos = .5, coords = "abs
```

makeTitle  
Creates an object of class Title

**Description**
Create an object of class Title which can be used to add a title to the plot

**Usage**

```r
makeTitle(text, cex, color, size)
```

**Arguments**
- `text`: The text that will make up the title
- `cex`: Font size of the title
- `color`: Font color of the title
- `size`: Size of the viewport in which the title resides
makeTranscript

Value

Object of class Title

Author(s)

Steffen Durinck and Jim Bullard

References

~put references to the literature/web site here ~

See Also

gdPlot

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 (text, cex, color, size)
{
  dp <- getClass("Title")@prototype@dp
  if (!missing(cex))
    setPar(dp, "cex", cex)
  if (!missing(color))
    setPar(dp, "color", color)
  if (!missing(size))
    setPar(dp, "size", size)
  new("Title", title = text, dp = dp)
}

makeTranscript

Creates an object of class Transcript

Description

Creates an object of class Transcript. This represents all known transcript structures in Ensembl.

Usage

makeTranscript(id, type, biomart, dp = NULL)

Arguments

id An identifier used to specify of which gene/transcript the transcript structures should be retrieved

type The type of identifiers used, examples are ensembl_gene_id, hgnc_symbol, entrezgene. See listAttributes function of thebiomaRt package for more info

biomart Mart object, created by the useMart function of biomaRt

dp object of class DisplayPars, determines the display of features on the plot
probestart

Contains dummy expression array probe start positions

Description

Contains dummy expression array probe start positions

Examples

#
**segEnd**

Contains dummy copy number segmentation end positions

**Description**

Contains dummy copy number segmentation end positions

**Examples**

```
#
```

---

**segStart**

Contains dummy copy number segmentation start positions data

**Description**

Contains dummy copy number segmentation start positions

**Examples**

```
#
```

---

**segments**

Contains dummy copy number segment data

**Description**

Contains dummy copy number segment data

**Examples**

```
#
```
seqDataEx  

This is an example data set from chromosome 4 of yeast from various publicly available datasets.

Description

This was a small dataset constructed from publicly available datasets. Please see references for details.

Usage

data(seqDataEx)

Format

data("seqDataEx", package = "GenomeGraphs") names(seqDataEx)

References

Ugrappa Nagalakshmi et. al. The transcriptional landscape of the yeast genome defined by RNA sequencing. Science, 2008
Adam Siepel, et. al. Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. Genome Res, 2005

Examples

data(seqDataEx)

setPar  

Sets a display parameter

Description

Sets a display parameter

Usage

setPar(obj, name, val, ...)

Arguments

obj An object, usually a gdObject.
name Name of display parameter to set.
val Value of display parameter.
... Ignored
showDisplayOptions  Print standard display options, DisplayPars for an object or a class

Description
Prints the available display options for a class or name of a class.

Usage
showDisplayOptions(obj, ...)

Arguments
obj Either an object of subclass gdObject or a character naming a class
...
~~Describe ... here~~

Value
Returns a DisplayPars object which is generally printed to the screen.

Examples
showDisplayOptions("GenericArray")

unrData  Contains exon array data

Description
Contains exon array data from the publically available dataset on human tissue panels, given by Affymetrix. The data was contributed to the package by Elizabeth Purdom.

Examples
#
unrNProbes  
Contents exon array data

**Description**
Contains the number of probes per exon array probeset id from the publically available dataset on human tissue panels, given by Affymetrix. The data was contributed to the package by Elizabeth Purdom.

**Examples**

```
#
```

unrPositions  
Contains probe start and end positions of exon array probes

**Description**
Contains probe start and end positions from the publically available dataset on human tissue panels, given by Affymetrix. The data was contributed to the package by Elizabeth Purdom.

**Examples**

```
#
```

eyeastCons1  
Contains dummy yeast conservation data

**Description**
Contains dummy yeast base conservation data.

**Examples**

```
#
```
Index

*Topic classes
  AnnotationTrack-class, 1
  BaseTrack-class, 2
  DisplayPars-class, 3
  ExonArray-class, 5
  gdObject-class, 23
  Gene-class, 6
  GeneModel-class, 7
  GeneRegion-class, 8
  GenericArray-class, 9
  GenomeAxis-class, 11
  HighlightRegion-class, 12
  Ideogram-class, 13
  Legend-class, 14
  Overlay-class, 15
  RectangleOverlay-class, 16
  Segmentable-class, 17
  Segmentation-class, 17
  TextOverlay-class, 18
  Title-class, 19
  Transcript-class, 19
  TranscriptRegion-class, 21

*Topic datasets
  cn, 22
  exonProbePos, 22
  ideogramTab, 27
  intensity, 27
  MappedRead-class, 15
  probestart, 42
  segEnd, 43
  segments, 43
  segStart, 43
  seqDataEx, 44
  unrData, 45
  unrNProbes, 46
  unrPositions, 46
  yeastCons1, 46

*Topic hplot
  DisplayPars, 4
  drawGD, 22
  gdPlot, 24
  geneBiomart, 25
  geneRegionBiomart, 25
  getPar, 26
  makeAnnotationTrack, 27
  makeBaseTrack, 28
  makeExonArray, 29
  makeGene, 31
  makeGeneModel, 32
  makeGeneRegion, 33
  makeGenericArray, 34
  makeGenericAxis, 35
  makeIdeogram, 36
  makeLegend, 37
  makeRectangleOverlay, 38
  makeSegmentation, 39
  makeTextOverlay, 40
  makeTitle, 40
  makeTranscript, 41
  setPar, 44
  showDisplayOptions, 45
  AnnotationTrack-class, 1
  BaseTrack-class, 2
  cn, 22
  DisplayPars, 4, 29, 32, 36
  DisplayPars-class, 3
  drawGD, 22
  drawGD, AnnotationTrack-method (AnnotationTrack-class), 1
  drawGD, BaseTrack-method (BaseTrack-class), 2
  drawGD, ExonArray-method (ExonArray-class), 5
  drawGD, Gene-method (Gene-class), 6
  drawGD, GeneModel-method (GeneModel-class), 7
  drawGD, GeneRegion-method (GeneRegion-class), 8
  drawGD, GenericArray-method (GenericArray-class), 9
  drawGD, GenomeAxis-method (GenomeAxis-class), 11
  drawGD, Ideogram-method (Ideogram-class), 13
INDEX

drawGD, Legend-method (Legend-class), 14
drawGD, MappedRead-method (MappedRead-class), 15
drawGD, Segmentation-method (Segmentation-class), 17
drawGD, Title-method (Title-class), 19
drawGD, Transcript-method (Transcript-class), 19
drawOverlay, TextOverlay-method (TextOverlay-class), 19
drawOverlay, RectangleOverlay-method (RectangleOverlay-class), 16

ExonArray-class, 5
exonProbePos, 22
gdObject, 1, 2, 5, 12, 15, 16, 18
gdObject-class, 23
gdPlot, 2, –6–11, 13, 14, 19–21, 24, 29–31, 33, 36–38, 41, 42
Gene-class, 6
geneBiomart, 25
GeneModel-class, 7
GeneRegion-class, 8
geneRegionBiomart, 25
GenericArray-class, 9
GenomeAxis-class, 11
getCex, gdObject-method (gdObject-class), 23
getColor, gdObject-method (gdObject-class), 23
getLty, gdObject-method (gdObject-class), 23
getLwd, gdObject-method (gdObject-class), 23
getPar, 26
getPar, DisplayPars-method (DisplayPars-class), 3
getPar, gdObject-method (gdObject-class), 23
getPch, gdObject-method (gdObject-class), 23
getPlotId, AnnotationTrack-method (AnnotationTrack-class), 1
getPointSize, gdObject-method (gdObject-class), 23
getSegmentation, Segmentable-method (Segmentable-class), 17
getSegmentStart, Segmentation-method (Segmentation-class), 17
getSegmentEnd, Segmentation-method (Segmentation-class), 17
getSize, gdObject-method (gdObject-class), 23
help, 35
HighlightRegion-class, 12
Ideogram-class, 13
ideogramTab, 27
initialize, AnnotationTrack-method (AnnotationTrack-class), 1
initialize, DisplayPars-method (DisplayPars-class), 3
initialize, gdObject-method (gdObject-class), 23
initialize, Gene-method (Gene-class), 6
initialize, GeneRegion-method (GeneRegion-class), 8
initialize, Transcript-method (Transcript-class), 19
intensity, 27
Legend-class, 14
makeAnnotationTrack, 27
makeBaseTrack, 28
makeExonArray, 29
makeGene, 31
makeGeneModel, 32
makeGeneRegion, 33
makeGenericArray, 34
makeGenomeAxis, 35
makeIdeogram, 36
makeLegend, 37
makeLegend, 37
makeRectangleOverlay, 16, 38
makeSegmentation, 39
makeTextOverlay, 18, 40
makeTitle, 40
makeTranscript, 41
MappedRead-class, 15
Overlay, 16, 18
Overlay-class, 15
probestart, 42
RectangleOverlay-class, 16
segEnd, 43
Segmentable-class, 17
INDEX

Segmentation, 17
Segmentation-class, 17
segments, 43
segStart, 43
seqDataEx, 44
setPar, 44
setPar, DisplayPars-method
  (DisplayPars-class), 3
setPar, gdObject-method
  (gdObject-class), 23
show, BaseTrack-method
  (BaseTrack-class), 2
show, DisplayPars-method
  (DisplayPars-class), 3
show, ExonArray-method
  (ExonArray-class), 5
show, Gene-method (Gene-class), 6
show, GeneRegion-method
  (GeneRegion-class), 8
show, GenericArray-method
  (GenericArray-class), 9
show, MappedRead-method
  (MappedRead-class), 15
show, Transcript-method
  (Transcript-class), 19
show, TranscriptRegion-method
  (TranscriptRegion-class), 21
showDisplayOptions, 45
showDisplayOptions, character-method
  (gdObject-class), 23
showDisplayOptions, gdObject-method
  (gdObject-class), 23

TextOverlay-class, 18
Title-class, 19
Transcript-class, 19
TranscriptRegion-class, 21

unrData, 45
unrNProbes, 46
unrPositions, 46

yeastCons1, 46