Tools for high throughput SNP chip data

Robert Scharpf

March 13, 2008

Introduction

SNPchip defines classes and methods useful for organizing high throughput genomic data. The classes defined here extend the `eSet` class in *Biobase*, utilizing the existing Bioconductor infrastructure for genomic data. This provides a foundation upon which statistical and visualization tools can be further developed. See (2) for additional details.

1 Simple Usage

```
> library(SNPchip)

> data(sample.snpset)
> sample.snpset

oligoSnpSet (storageMode: lockedEnvironment)
assayData: 5859 features, 5 samples
  element names: calls, callsConfidence, cnConfidence, copyNumber
experimentData: use 'experimentData(object)'
Annotation: pd.mapping50k.xba240
phenoData
An object of class "AnnotatedDataFrame"
  sampleNames: NA17101_X_hAF_A1_4000091.CEL, NA17102_X_hAF_A
  2_4000091.CEL, ..., NA17105_X_hAF_A5_4000091.CEL  (5 total)
  varLabels and varMetadata description: none
featureData
An object of class "AnnotatedDataFrame"
  rowNames: SNP_A-1507972, SNP_A-1641761, ..., SNP_A-1759046  (5859 total)
  varLabels and varMetadata description:
    dbsnp_rs_id: dbsnp_rs_id
    chrom: chrom
    ...: ...
    enzyme: enzyme
    (8 total)
Annotation [1] "pd.mapping50k.xba240"

> fD <- new("AnnotatedDataFrame", data = data.frame(row.names = featureNames(sample.snpset)),
+     varMetadata = data.frame(labelDescription = character()))
> featureData(sample.snpset) <- fD
```

For Affymetrix platforms, access to SNP-level annotation such as chromosome and physical position is through *RSQLite*. See the *oligo* vignette for additional information regarding available SNP-level Annotation. Accessing RSQLite databases each time chromosome and physical position are needed (such as for making scatterplots of physical position versus copy number) may increase the time of computation, particularly for small objects such as the `sample.snpset`. Therefore, one may wish to add this information to the `featureData` slot. Subsequent calls to chromosome and physical position will look in the `featureData` before searching in the RSQLite database. Because `sample.snpset` is small and adding this annotation to the `featureData` slot will greatly increase the speed for graphing the data (the graphs make repeated use of chromosome and position calls), we add this annotation to the `featureData` for this vignette.

```
> if (require("pd.mapping50k.xba240")) {
+     system.time(tmp <- chromosome(sample.snpset))
+     object.size(sample.snpset)
+     featureData(sample.snpset)$chromosome <- chromosome(sample.snpset)
+     featureData(sample.snpset)$position <- position(sample.snpset)
+     system.time(tmp <- chromosome(sample.snpset))
+     object.size(sample.snpset)
+ }
```

```
[1] 312932
```

Note that if one is using a platform for which there is no `pd.mapping` package available, one must provide the chromosome (character string) and physical position (integer) in the `featureData` slot using the column labels "chromosome" and "position", respectively.
Subsetting an object inheriting from the class `SnpLevelSet` is done in the usual way. For instance, here we select SNPs on the first three chromosomes and then generate an object of class `ParSnpSet` that contains graphical parameters for visualizing the data for this subset:

```
> snpset <- sample.snpset[chromosome(sample.snpset) %in%
+     as.character(1:3), 1:4]
> graph.par <- new("ParSnpSet")
> graph.par$use.chromosome.size <- TRUE
> graph.par$cex <- 3
> graph.par <- getPar(graph.par, snpset)
```

```
[1] "one.ylim is FALSE. Calculating ylim based on the percentiles of the copy number distribution"
```

```
> graph.par$ylim <- c(0.7, 8)
> class(graph.par)
```

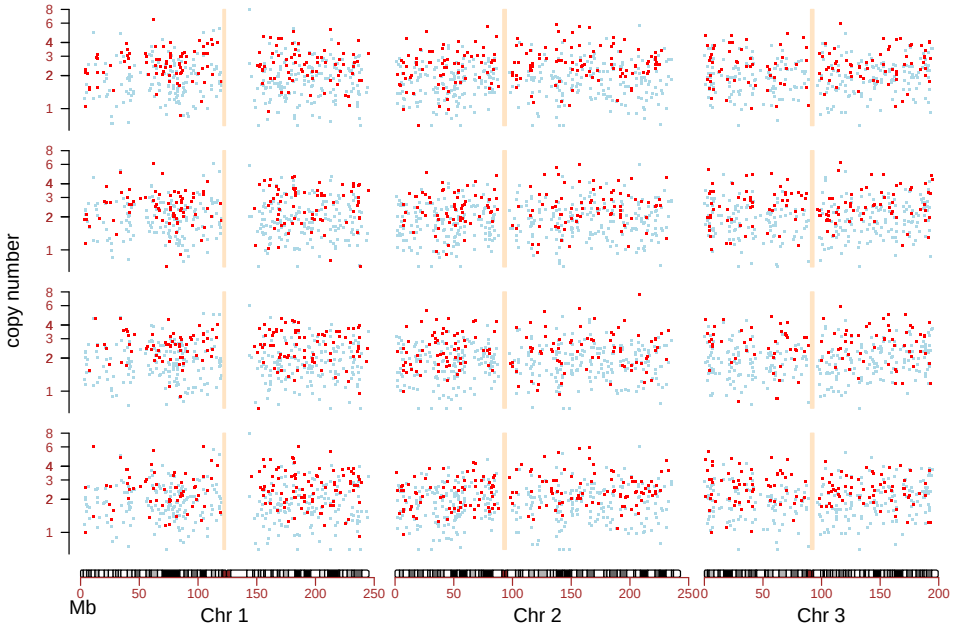
```
[1] "ParSnpSet"
attr(,"package")
[1] "SNPchip"
```

Plot the first few chromosomes for samples 1-4:

```
> plotSnp(graph.par, snpset)
```

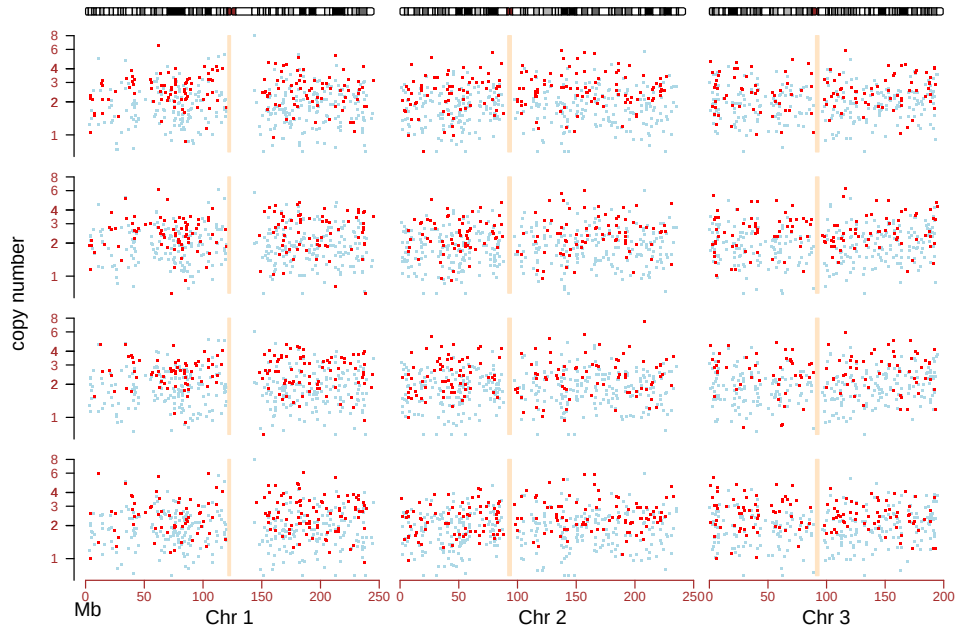
```
Object of class  ParSnpSet
$col.axis
[1] "brown"
```

```
$cex.main
[1] 1

$cex.axis
[1] 1

$cex.legend
[1] 1

$cex.lab
[1] 1

...
```




```
> graph.par$cytoband.side <- 3
> graph.par$heights <- rev(graph.par$heights)

> plotSnp(graph.par, snpset)

Object of class  ParSnpSet
$col.axis
[1] "brown"

$cex.main
[1] 1

$cex.axis
[1] 1
```

```
$cex.legend
[1] 1

$cex.lab
[1] 1

...
```



The samples are plotted by row. For each sample, the copy number (vertical axis) is plotted against the physical position of the SNP in the chromosome. Here, the chromosome labels are plotted beneath the cytobands.

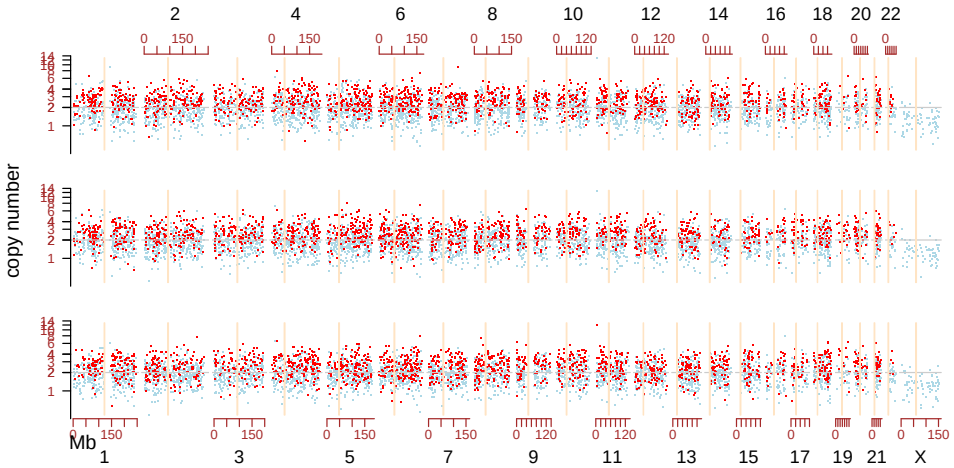
2 Examples

2.1 Genome-wide plots for multiple samples

A genome-wide view of copy number and genotype calls versus physical position can be made using `plotSnp`. Here, we plot chromosomes 1-22 and X of samples 1 - 3 in the object `sample.snpset`:

```
> graph.par$mar <- c(0.1, 0.1, 2, 0.1)
> graph.par$oma <- c(3, 4, 2, 1)
> graph.par$cex <- 2
> graph.par$abline <- TRUE
> graph.par$cex.lab <- 0.9
> graph.par <- getPar(graph.par, sample.snpset[,
+     1:3], add.cytoband = FALSE)

[1] "one.ylim is FALSE. Calculating ylim based on the percentiles of the copy number distribution"

> plotSnp(graph.par, sample.snpset[, 1:3])
```

```
Object of class  ParSnpSet
$col.axis
[1] "brown"

$cex.main
[1] 1

$cex.axis
[1] 1

$cex.legend
[1] 1

$cex.lab
[1] 0.9

...
```



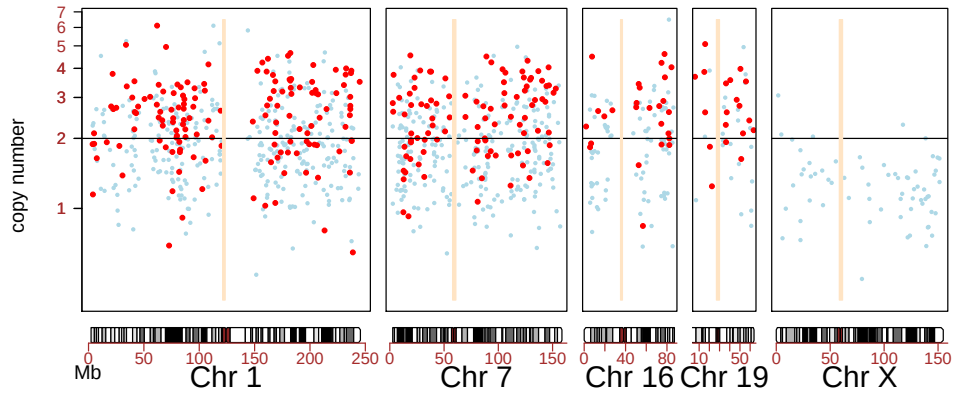
Note that we suppress the cytobands in the above plot (the resolution is too poor at this level) by the argument `add.cytoband` in the function `getPar`. The default plot layout generally works well, but can be adjusted through additional arguments to par and layout.

2.2 Subsetting for more focused plots

A more focused view of chromosomes 1, 7, 16, 19, and X of sample 2 could be obtained by

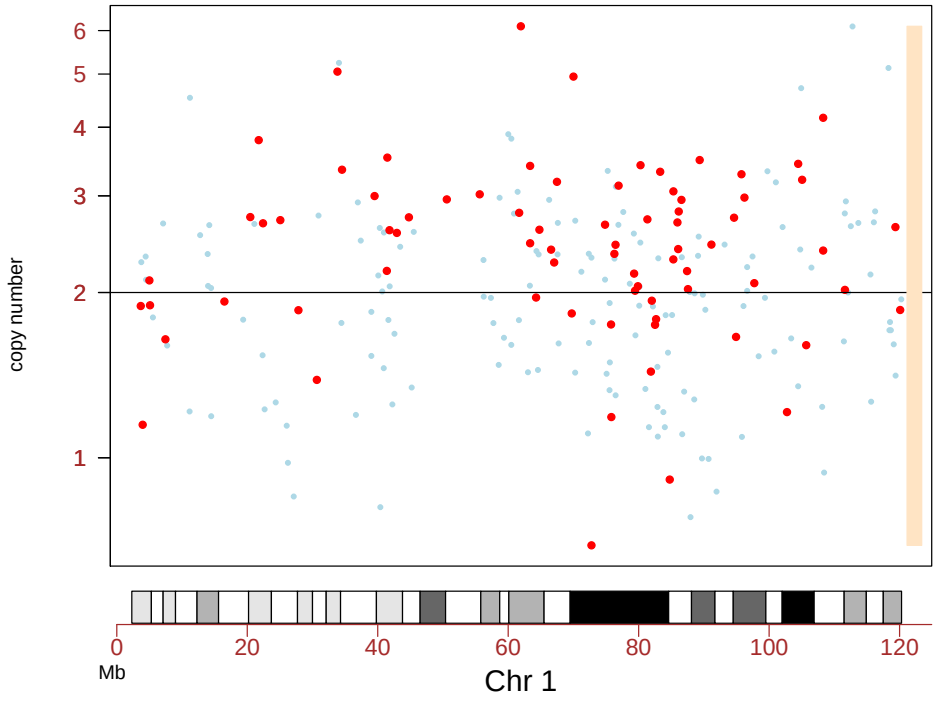
```
> graph.par <- new("ParSnpSet")
> graph.par <- getPar(graph.par, sample.snpset[chromosome(sample.snpset) %in%
+     c(1, 7, 16, 19, "X"), 2])

[1] "one.ylim is FALSE. Calculating ylim based on the percentiles of the copy number distribution"

> graph.par$cex <- 0.8
> graph.par$mar <- rep(0.5, 4)
> graph.par$pch <- c(20, 21, 20)
> graph.par$bty <- "o"
> graph.par$cex.axis <- 1.2
```

```
> graph.par$cex.lab <- 1.5
> graph.par$xaxs <- "r"
> graph.par$abline <- TRUE
> graph.par$abline.col <- "black"

> plotSnp(graph.par, sample.snpset[chromosome(sample.snpset) %in%
+     c(1, 7, 16, 19, "X"), 2])

Object of class  ParSnpSet
$col.axis
[1] "brown"

$cex.main
[1] 1

$cex.axis
[1] 1.2

$cex.legend
[1] 1

$cex.lab
[1] 1.5

...
```



A plot of just the p-arm in sample 2 of chromosome 1.

```
> data(chromosomeAnnotation)
> parm <- chromosomeAnnotation["1", "centromereStart"]
> snpset <- sample.snpset[chromosome(sample.snpset) ==
+     "1" & position(sample.snpset) < parm, 2]
> graph.par$use.chromosome.size <- FALSE
> graph.par <- getPar(graph.par, snpset)

> plotSnp(graph.par, snpset)

Object of class  ParSnpSet
$col.axis
[1] "brown"

$cex.main
[1] 1

$cex.axis
[1] 1.2

$cex.legend
[1] 1

$cex.lab
[1] 1.5

...
```

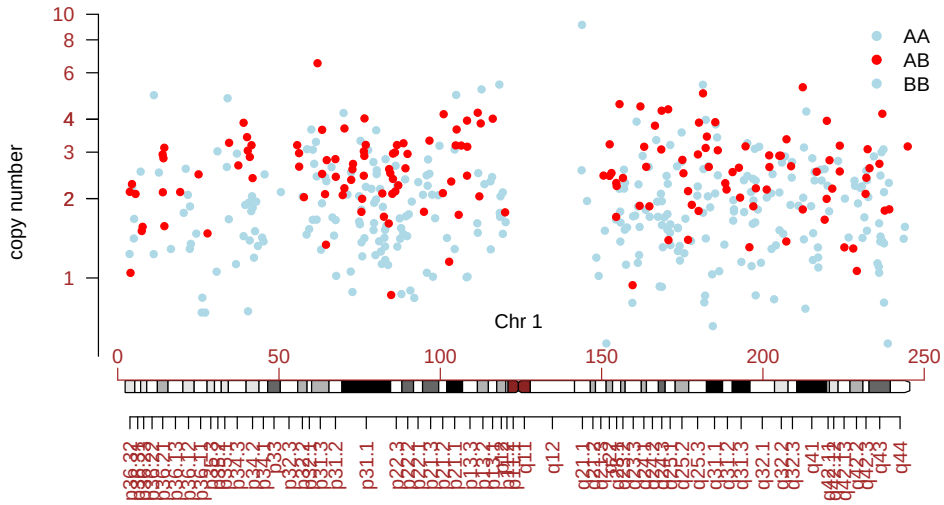


Note that the cytoband is automatically subsetted appropriately. Had we instead specified `use.chromosome.size=TRUE`, the x-axis limits would include the entire chromosome (and cytoband) though only the SNPs on the p-arm would be plotted.
Adding a legend for the genotypes

```
> data(sample.snpset)
> x <- sample.snpset[chromosome(sample.snpset) ==
+     "1", 1]
> gp <- new("ParSnpSet")
> gp <- getPar(gp, x)
> gp$legend <- c("AA", "AB", "BB")
> gp$legend.col <- gp$col
> gp$legend.bg <- gp$bg
> gp$pch <- 21
> gp$cex <- 0.8
> gp$label.cytoband <- TRUE
> gp$add.centromere <- FALSE
> gp$xlab <- ""
> plotSnp(gp, x)
```

```
Object of class  ParSnpSet
$col.axis
[1] "brown"

$cex.main
[1] 1
```

```
$cex.axis
[1] 1

$cex.legend
[1] 1

$cex.lab
[1] 1

...
```



Alternatively, plot the cytoband on top and then add the legend to the plot by hand.

```
> gp <- new("ParSnpSet")
> gp <- getPar(gp, x)
> gp$pch <- 21
> gp$cex <- 0.8
> gp$cytoband.side <- 3
> gp$heights <- rev(gp$heights)
> gp$label.cytoband <- TRUE
> gp$add.centromere <- FALSE
> plotSnp(gp, x)

Object of class  ParSnpSet
$col.axis
[1] "brown"

$cex.main
[1] 1

$cex.axis
[1] 1
```

```
$cex.legend
[1] 1

$cex.lab
[1] 1

...
```

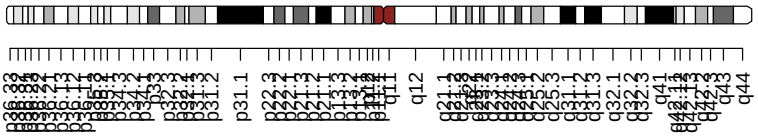
```
> legend("bottomleft", pch = 21, col = gp$col, pt.bg = gp$bg,
+     legend = c("AA", "AB", "BB"), bty = "n")
```

2.3 Plotting cytoband

To plot the cytoband of chromosome 1,

```
> plotCytoband("1")
```

```
NULL
```



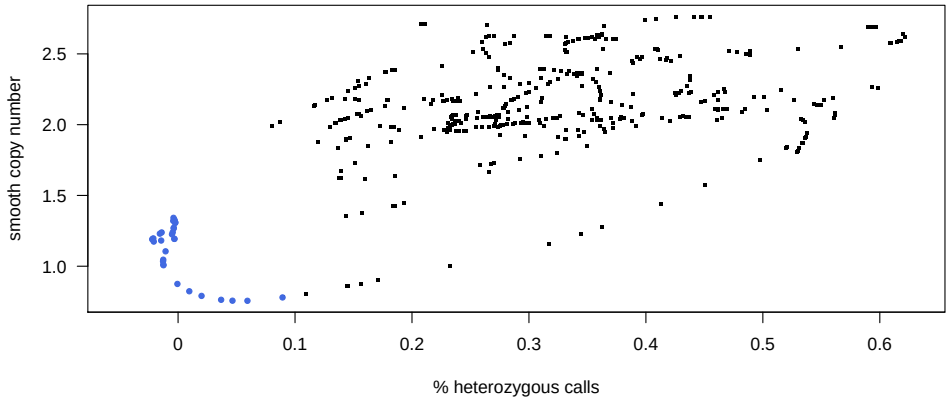
2.4 Smoothing example

Hidden Markov models for objects of class `SnpCallSet`, `SnpCopyNumberSet`, and `oligoSnpSet` are available in the package *VanillaICE* available at Bioconductor. A more detailed description of the hidden Markov models fit in the *VanillaICE* package are discussed elsewhere (1). Here we provide an example of a lowess smoother for copy number estimates. The following code chunk first assigns heterozygous calls to the integer 1 and homozygous calls to the integer zero. It follows that regions of deletions will have homozygous calls of zero. We simulated a deletion of 50 consecutive SNPs and then converted the `sample.snpset` to a list where each element in the list is an `oligoSnpSet` object for one chromosome.

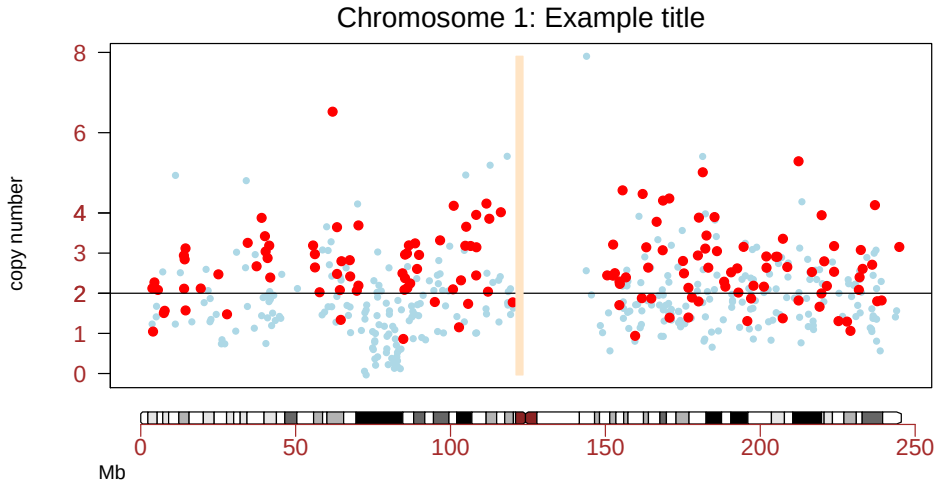
```
> sim1 <- sample.snpset[chromosome(sample.snpset) %in%
+     1:5, 1]
> sim1 <- sim1[chromosome(sim1) == "1", ]
> sim1 <- sim1[order(position(sim1)), ]
> copyNumber(sim1)[101:150, 1] <- copyNumber(sim1)[101:150,
+     1] - 1
> calls(sim1)[101:150, 1] <- 1
> smoothSet <- smoothSnp(sim1, 1:5, 1:3, span = 1/10)
> highlight <- calls(smoothSet)[, 1] <= 0.1 & copyNumber(smoothSet)[,
+     1] <= 1.5
```

A plot of the smoothed calls versus copynumber can be used to visualize the deletion:

```
> op <- par(las = 1, mfrow = c(1, 1), mar = c(5,
+     4, 0.5, 0.5), oma = rep(0, 4))
> plot(calls(smoothSet)[, 1], copyNumber(smoothSet)[,
+     1], ylim = range(copyNumber(smoothSet)), pch = ".",
+     cex = 3, xlab = "% heterozygous calls", ylab = "smooth copy number",
+     xaxt = "n", xlim = c(-0.05, 30/70 + 0.2))
> axis(1, at = pretty(calls(smoothSet)), labels = pretty(calls(smoothSet)))
> points(calls(smoothSet)[highlight, 1], copyNumber(smoothSet)[highlight,
+     1], pch = 20, col = "royalblue", bg = "white")
> par(op)
```




```
> graph.par$cex <- 1
> graph.par$use.chromosome.size <- TRUE
> graph.par$main <- "Chromosome 1: Example title"
> graph.par <- getPar(graph.par, sim1)
> graph.par$label.chromosome <- FALSE
> plotSnp(graph.par, sim1)

Object of class  ParSnpSet
$col.axis
[1] "brown"

$cex.main
[1] 1

$cex.axis
[1] 1.2

$cex.legend
[1] 1

$cex.lab
[1] 1.5

...
```



2.5 Descriptive and statistical summaries

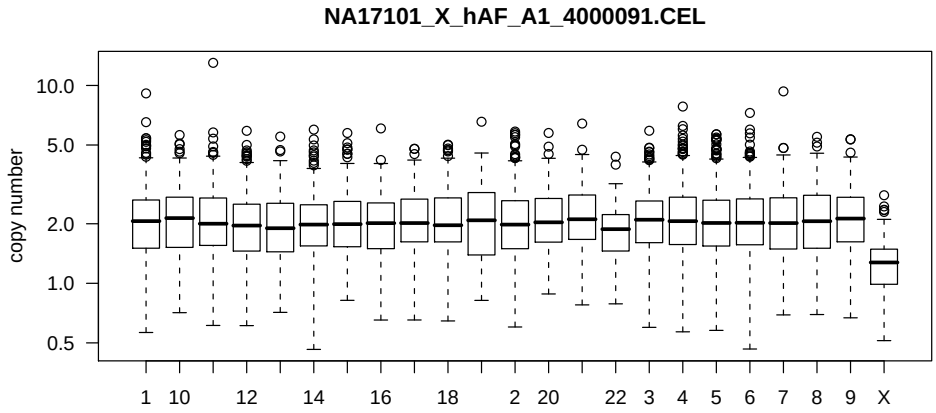
Descriptive statistics for copy number and genotype calls are provided with the `summary` method. For each chromosome in the `oligoSnpSet`, `summary` calculates the average and standard deviation of the copy number estimates, as well as the % homozygous and heterozygous calls. In addition, summary calculates the average copy number, standard deviation, % homozygous and heterozygous across all autosomes in the `oligoSnpSet`. The dimensions of the four matrices are S x C + 1, where S is the number of samples and C is the number of chromosomes in the `oligoSnpSet`.

```
> x <- summary(sample.snpset, digits = 1)
> str(x)

List of 5
 $ avg.CN     : num [1:23, 1:5] 2.2 2.2 2.2 2.1 2 2.1 2.2 2.1 2.2 2.2 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:23] "1" "10" "11" "12" ...
  .. ..$ : chr [1:5] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
 $ sd.CN      : num [1:23, 1:5] 1 0.9 1.1 0.8 0.8 0.9 0.9 0.8 0.8 0.9 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:23] "1" "10" "11" "12" ...
  .. ..$ : chr [1:5] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
 $ prop.Hom   : num [1:23, 1:5] 0.7 0.7 0.7 0.6 0.7 0.7 0.7 0.7 0.7 0.7 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:23] "1" "10" "11" "12" ...
  .. ..$ : chr [1:5] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
 $ prop.Het   : num [1:23, 1:5] 0.3 0.3 0.3 0.4 0.3 0.3 0.3 0.3 0.3 0.3 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:23] "1" "10" "11" "12" ...
  .. ..$ : chr [1:5] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
 $ prop.NoCall: num [1:23, 1:5] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
```

```
  .. ..$ : chr [1:23] "1" "10" "11" "12" ...
  .. ..$ : chr [1:5] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
```

Boxplot by chromosome:

```
> op <- par(mfrow = c(1, 1), mar = c(4, 4, 3, 1),
+     las = 1)
> boxplot(split(copyNumber(sample.snpset[, 1]),
+     chromosome(sample.snpset)), ylab = "copy number",
+     main = sampleNames(sample.snpset)[1], log = "y")
> par(op)
```



3 Annotation

3.1 Chromosome-level

The chromosome-level annotation used in the plotting methods can be accessed by `data()` calls:

```
> data(chromosomeAnnotation)
> chromosomeAnnotation[1:5, ]
```

```
  centromereStart centromereEnd chromosomeSize
1       121147476     123387476      245522847
2        91748045      94748045      243018229
3        90587544      93487544      199505740
4        49501045      52501045      191411218
5        46441398      49441398      180857866
```

```
> data(cytoband)
> cytoband[1:5, ]
```

```
  chrom chromStart chromEnd   name gieStain
1     1          0  2300000 p36.33     gneg
2     1    2300000  5300000 p36.32   gpos25
3     1    5300000  7100000 p36.31     gneg
4     1    7100000  9000000 p36.23   gpos25
5     1    9000000 12300000 p36.22     gneg
```

This is a static table and will be deprecated once chromosome-level annotation is provided by Bioconductor. If chromosome-level annotation is available in Bioconductor, please send me a note.

3.2 Feature-level

Feature-level annotation for Affymetrix platforms is available in the pd.mapping packages.
See the *oligo* vignette for additional information about available feature-level annotation.

4 Integration with other Bioconductor packages

4.1 *oligo*

For generating `SnpCallSets` from .CEL files, see the R package *oligo*. In particular, the function `crlmm` in *oligo* creates an instance of the class `SnpCallSet`. A `oligoSnpSet` can be created if the the copy number estimates are obtained by some other means.

4.2 *RSNPper*

To retreive additional annotation on the known SNP's in the region of this simulated deletion, we could use the *RSNPper*.

```
> library(RSNPper)
> (dbId <- dbSnpId(annSnpset)[snps[2] == featureNames(annSnpset)])
> dbId <- strsplit(dbId, "rs")[[1]][2]
> print(SNPinfo(dbId))
```

5 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 2.7.0 Under development (unstable) (2008-01-28 r44219), `powerpc-apple-darwin8.11.0`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, splines, stats, tools, utils
- Other packages: AnnotationDbi 0.99.23, Biobase 1.99.0, DBI 0.2-3, RSQLite 0.6-4, SNPchip 1.3.18, affxparser 1.9.5, genefilter 1.15.10, oligo 1.3.15, oligoClasses 1.1.15, pd.mapping50k.xba240 0.3.4, preprocessCore 0.99.22, survival 2.32
- Loaded via a namespace (and not attached): annotate 1.15.6

References

[1] Robert B Scharpf, Giovanni Parmigiani, Jonathan Pevsner, and Ingo Ruczinski. Hidden Markov models for the assessment of chromosomal alterations using high-throughput SNP arrays. *Annals of Applied Statistics*, 2008. To appear.

[2] Robert B Scharpf, Jason C Ting, Jonathan Pevsner, and Ingo Ruczinski. SNPchip: R classes and methods for SNP array data. *Bioinformatics*, 23(5):627–628, Mar 2007.