Tools for high throughput SNP chip data

Robert Scharpf, Jonathan Pevsner, Jason Ting, and Ingo Ruczinski

September 14, 2007

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 organizing high dimensional genomic data. This provides a foundation upon which statistical and visualization tools can be further developed.

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"
```

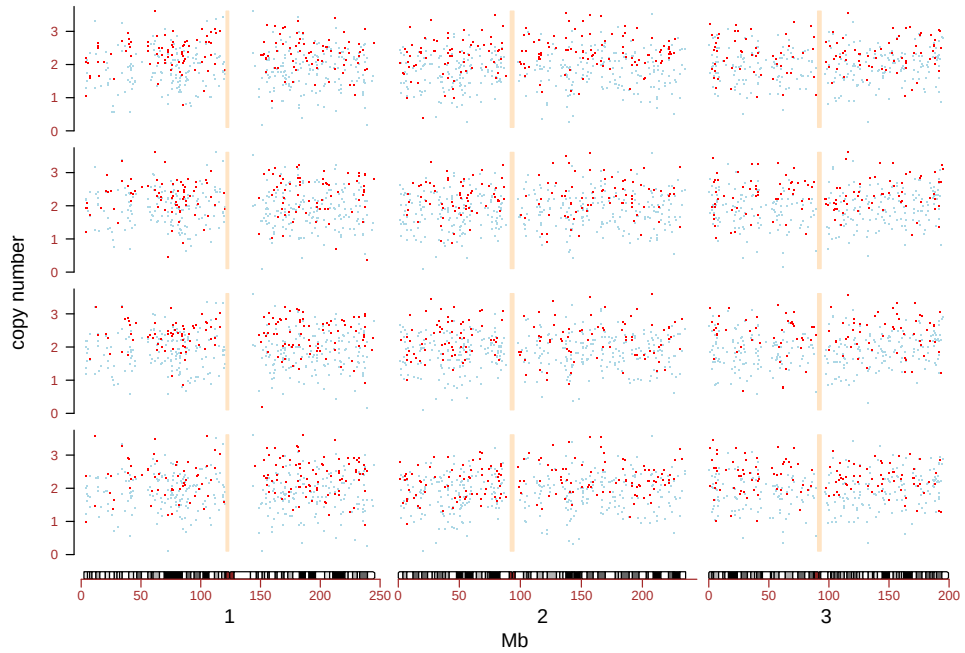
```
> snpset <- sample.snpset[chromosome(sample.snpset) %in%
+     as.character(1:3), 1:4]
> graph.par <- getPar(snpset)
```

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

```
> class(graph.par)

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

> graph.par$label.cytoband <- FALSE
> graph.par$use.chromosome.size <- TRUE
```

Plot the first few chromosomes for samples 1-4:

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

NULL
```



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.

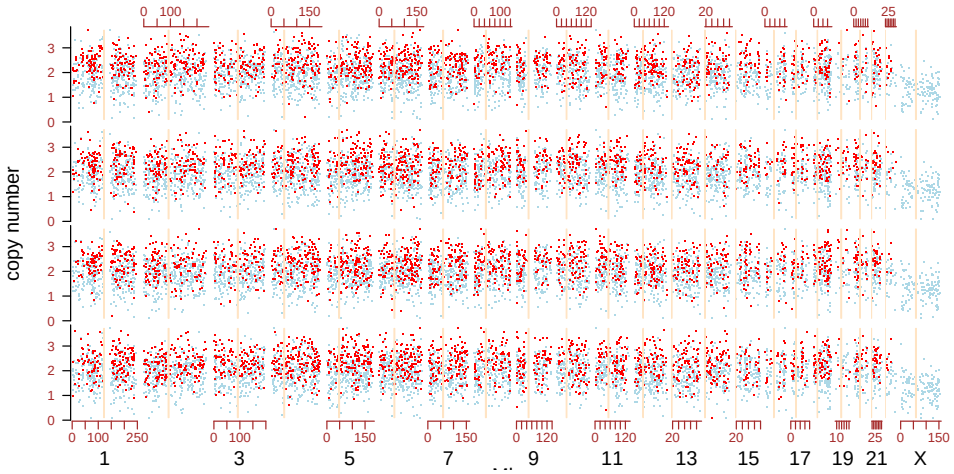
2 Examples

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 - 4 in the object `sample.snpset`:

```
> graph.par <- getPar(sample.snpset[, 1:4], add.cytoband = FALSE)

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

```
> graph.par$cex <- 2
> graph.par$use.chromosome.size <- TRUE
> graph.par$mar <- rep(0.1, 4)
> graph.par$oma <- c(3, 4, 2, 1)

> plotSnp(graph.par, sample.snpset[, 1:4])

NULL
```

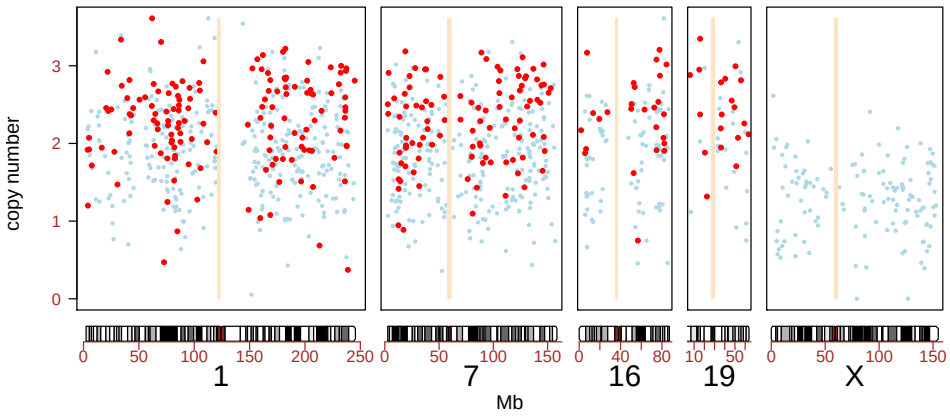


Note that we suppress the cytobands in the above plot (the resolution is too poor at this level) by the argument `add.cytoband` in the function `getPar`. The default plot layout generally works well, but can be adjusted through additional arguments to par and layout.

2.2 Subsetting for more-focused plots

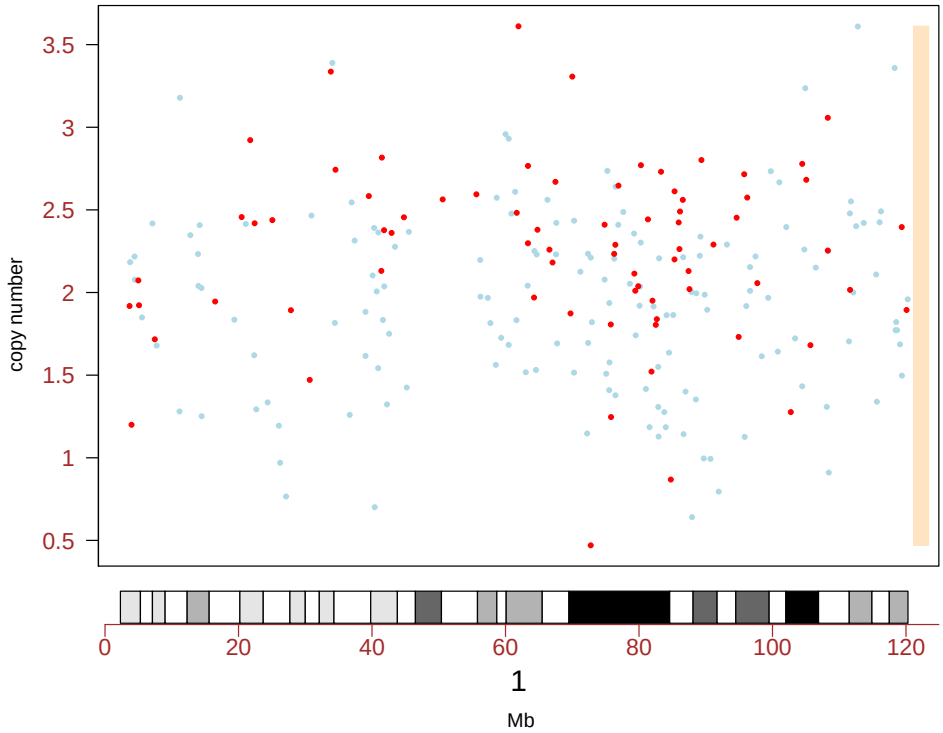
A more focused view of chromosomes 1, 7, 16, 19, and X of sample 2 could be obtained by

```
> graph.par <- getPar(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$xaxs <- "r"
> graph.par$use.chromosome.size <- TRUE

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

NULL
```



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 <- getPar(snpset)
> graph.par$cex <- 0.8
> graph.par$pch <- 20
> graph.par$bg <- c("royalblue", "red", "royalblue")
> graph.par$bty <- "o"
> graph.par$cex.axis <- 1.2
> graph.par$cex.lab <- 1.5

> plotSnp(graph.par, snpset)

NULL
```

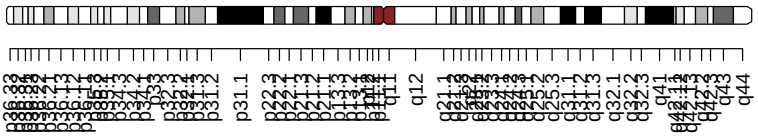


Note that the cytoband is automatically subsetted appropriately. Had we instead specified `use.chromosome.size=TRUE`, the x-axis limits would include the entire chromosome (and cytoband) though only the SNPs on the p-arm would be plotted.

2.3 Plotting cytoband

To plot the cytoband of chromosome 1,

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



2.4 Smoothing example

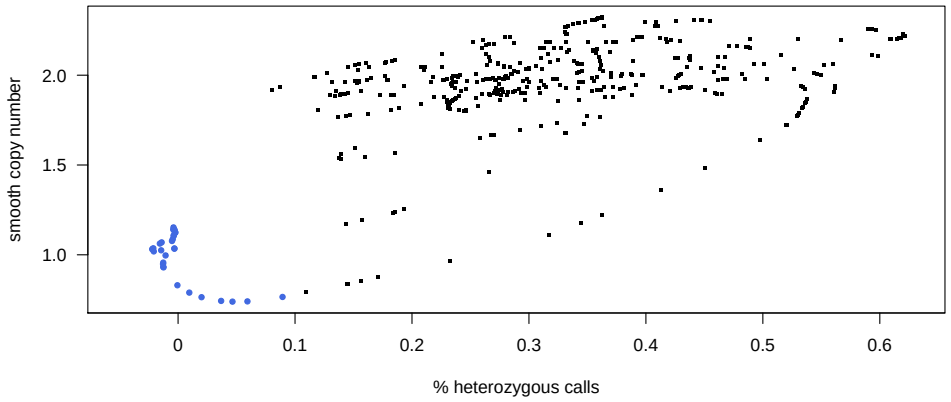
Here we discuss a quick method for smoothing copy number estimates for each chromosome. Better smoothing of the copy number estimates can be acheived by hidden Markov models. The following code chunk first assigns heterozygous calls to the integer 1 and homozygous calls to the integer zero. It follows 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
```

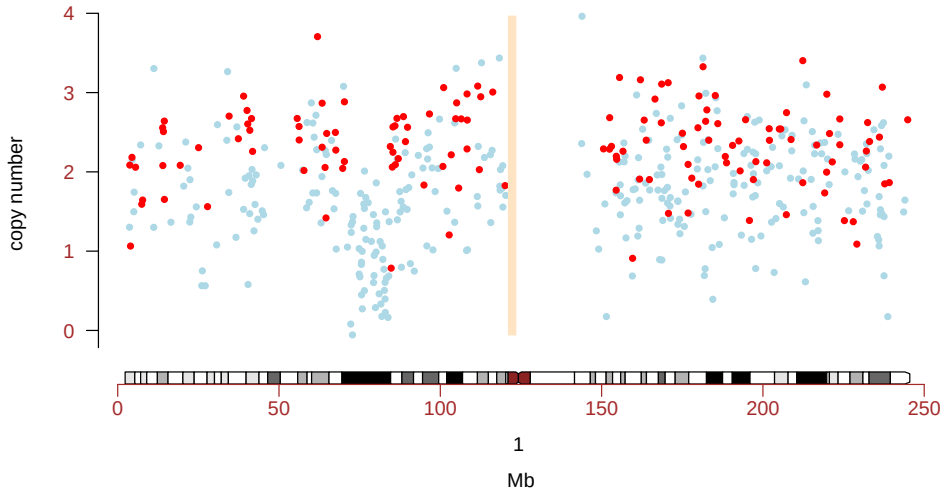
A plot of the smoothed calls versus copynumber can be used to visualize the deletion:

```
> 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 <- getPar(sim1)
> graph.par$cex <- 1
> graph.par$pch <- 20
> graph.par$use.chromosome.size <- TRUE
> graph.par$main <- "Chromosome 1"
> plotSnp(graph.par, sim1)
```

```
NULL
```



2.5 Descriptive and statistical summaries

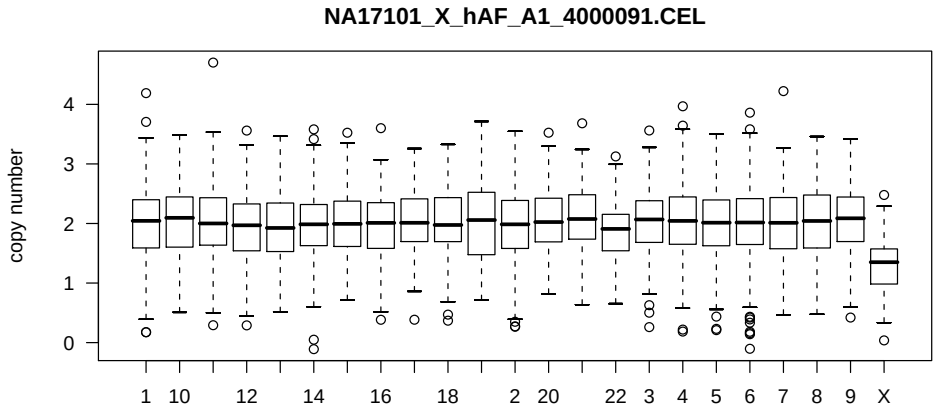
Descriptive statistics for copy number and genotype calls are provided with the `summary` method. For each chromosome in the `oligoSnpSet`, `summary` calculates the average and standard deviation of the copy number estimates, as well as the % homozygous and heterozygous calls. In addition, summary calculates 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 1.9 2 2 1.9 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] 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.5 0.6 ...
  ..- 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])
> 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
```

3.2 Feature-level

For ease of subsetting with the plotting routines, we currently store the feature-level annotation in the `featureData` slot. This can be acheived by
Alternatively, one may obtain the NetAffx annotation saved as an R object here:

```
> path <- "http://biostat.jhsph.edu/~iruczins/publications/sm/2006.scharpf.bioinfo"
> try(load(url(paste(path, "/mapping/mapping10k.rda",
+     sep = ""))))
> colnames(mapping10k$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 2.6.0 alpha (2007-09-05 r42788), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8;LC_NUMERIC=C;LC_TIME=en_US.UTF-8;LC_COLLATE=en_US.UTF-8;LC_MONETARY=en_US.UTF-8;LC_MESSAGES=en_US.UTF-8;LC_PAPER=en_US.UTF-8;LC_NAME=C;LC_ADDRESS=C;LC_T8;LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, splines, stats, tools, utils
- Other packages: Biobase 1.15.33, genefilter 1.13.12, oligoClasses 0.99.1, SNPchip 1.1.29, survival 2.32
- Loaded via a namespace (and not attached): annotate 1.15.3, AnnotationDbi 0.0.88, DBI 0.2-3, RSQLite 0.6-0