Tools for high throughput SNP chip data

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

July 23, 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

We illustrate the structure of the class `AnnotatedSnpSet` with a small dataset provided with the package.

```
> options(width = 69)
> library(SNPchip)
> data(annSnpset)
> annSnpset

AnnotatedSnpSet (storageMode: lockedEnvironment)
assayData: 5896 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_A2_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  (5896 total)
  varLabels and varMetadata description:
    dbsnp_rs_id: dbsnp_rs_id
    chrom: chrom
    ...: ...
    enzyme: enzyme
    (8 total)
Annotation [1] "pd.mapping50k.xba240"

chromosomeAnnotation
    centromereStart centromereEnd chromosomeSize
1         121147476     123387476      245522847
```

```
2          91748045       94748045       243018229
...             ...            ...             ...
Y          11237315       12237315        57701691
```

`annSnpset` is an instance of the `AnnotatedSnpSet` class. Here, the assayData slot in `AnnotatedSnpSet` contains 5896 SNPs with estimates of copy number and genotype calls, as well as a corresponding confidence score. Typically, such an object would contain 100,000 - 500,000 estimates of genotype calls and copy number. We illustrate in Section 3 how to create an instance of `AnnotatedSnpSet` from probe-level summaries of SNP chip data. In addition to estimates of genotype call and copy number, the `annSnpset` contains both chromosome-level annotation, as well as SNP-level annotation. The chromosome-level annotation includes the centromere start and stop sites and chromosome size (in number of base pairs), and could be extended to include location of cytobands, or any other feature of a chromosome.

```
> 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
```

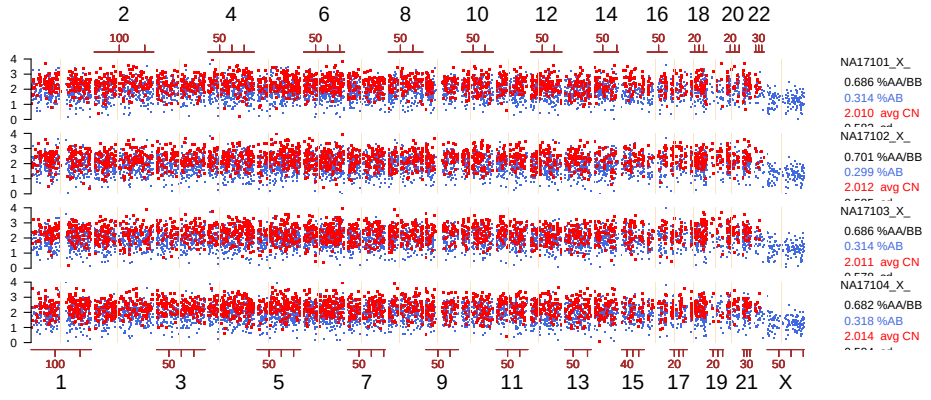
The method `getSnpAnnotation` retrieves SNP-level annotation from annotation packages maintained at Bioconductor. A more rich annotation is in development. The appropriate pd.mapping library needs to be loaded for this method to work.

```
> annotation(annSnpset)
> library("pd.mapping50k.xba240")
```

```
> featureData(annSnpset) <- getSnpAnnotation(annSnpset)
```

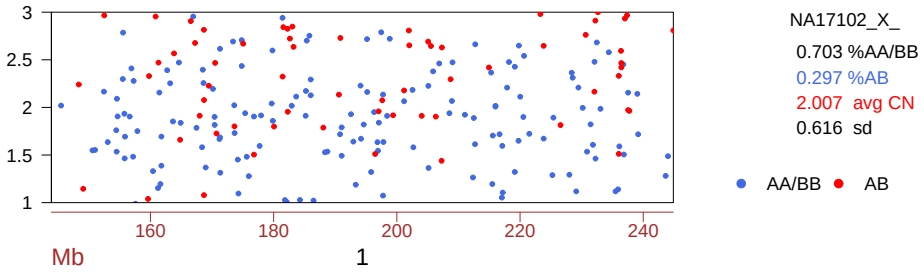
A genome-wide view of copy number and genotype calls versus physical position can be made using `plotSnp`. Here, we plot chromosomes 1-22 and X (the integer 23 is used to represent X) of samples 1 - 4 in the object `annSnpset`:

```
> plotSnp(annSnpset, chromosomes = c(1:22, "X"), samples = 1:4,
+     width.right = 10, cex.axis = 1, lab = c(3, 3, 5),
+     cex.lab = 1.2, cex.legend = c(1, 1.2), legend = c(TRUE,
+         FALSE))
```



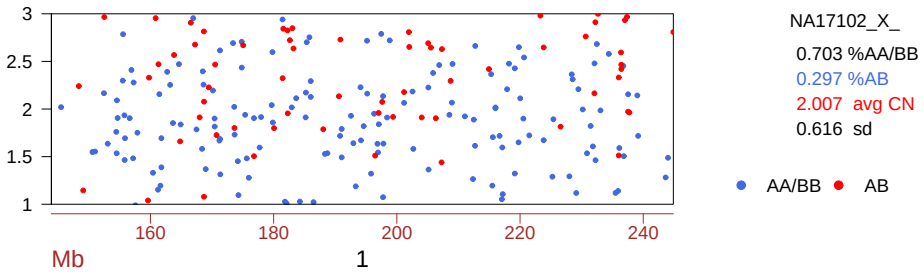
The copy number estimates have been centered to have mean zero – a centered copy number of 0 corresponds to a copy number of two. The default plot layout generally works well, but can be adjusted through the arguments mar, oma, and width.right in `plotSnp`. The latter argument specifies how much room to allow for the summary panel relative to the size of the smallest chromosome plotted. For instance, if plotting chromosomes 1-22 and X, width.right set to 15 allows a plotting region for the summary paenl that is 15 times larger than chromosome 21. A more focused view of chromosomes 1, 7, 16, 19, and X of sample 2 could be obtained by

```
> plotSnp(annSnpset, c(1, 7, 16, 19, "X"), c(2, 5), cex = c(1,
+     1, 1), pch = c(20, 21, 20), bg = c("royalblue",
+     "white", "royalblue"), bty = "o", width.right = 1.6,
+     cex.axis = 1.2, cex.lab = 1.5, cex.legend = c(1.2,
+         1.2), xaxs = "r")
```



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

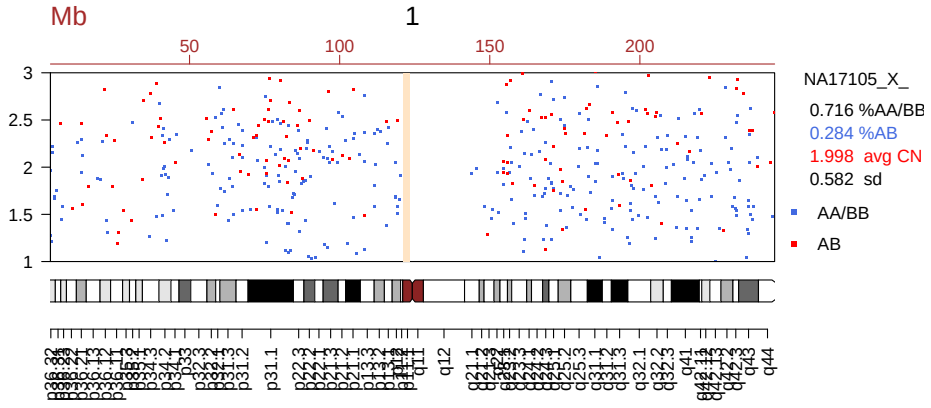
```
> chr1 <- annSnpset[chromosome(annSnpset) == "1", ]
> start <- min(position(chr1)[position(chr1) > chromosomeAnnotation["1",
+     1]], na.rm = TRUE)
> plotSnp(chr1[position(chr1) > start, ], 1, 2, xlim = range(position(chr1)[position(chr1) >
+     start], na.rm = TRUE), cex = c(1, 1, 1), pch = c(20,
+     20, 20), bg = c("royalblue", "red", "royalblue"),
+     bty = "o", width.right = 0.4, cex.axis = 1.2, cex.lab = 1.5,
+     cex.legend = c(1.2, 1.2))
```



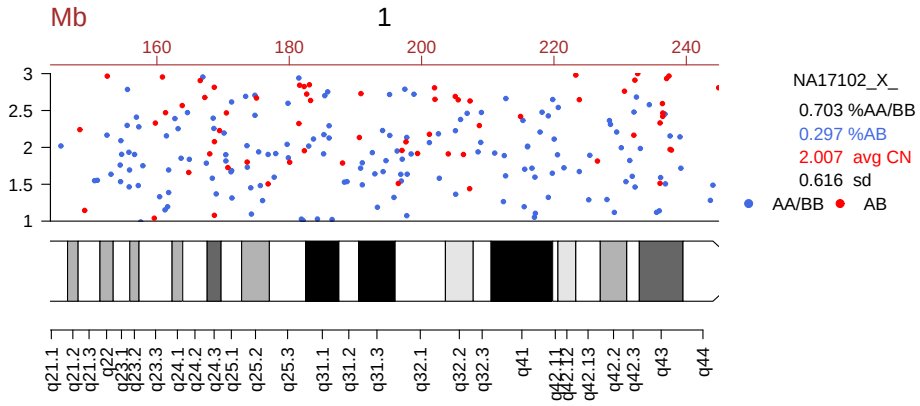
Cytobands can be added to graphs as follows:

```
> data(annSnpset)
> data(cytoband)
```

```
> chr1 <- annSnpset[chromosome(annSnpset) == "1", ]
> plotSnp(chr1, 1, 5, oma = c(6, 4, 0, 3), cex = c(3,
+     3, 3), cex.axis = 1.2, cex.legend = c(1.2, 1.2),
+     addCytoband = TRUE, legend.location = c("topleft",
+         "bottomleft"), height.cytoband = 0.2, width.right = 0.2,
+     bty = "o", cex.lab = 1.5, ncol = 1, adj = 0)
```




```
> plotSnp(chr1[position(chr1) > start, ], 1, 2, xlim = range(position(chr1)[position(chr1) >
+     start], na.rm = TRUE), cex = c(1, 1, 1), pch = c(20,
+     20, 20), bg = c("royalblue", "red", "royalblue"),
+     bty = "l", width.right = 0.3, cex.axis = 1.2, cex.lab = 1.5,
+     cex.legend = c(1.2, 1.2), addCytoband = TRUE)
```



2 Available annotation

Bioconductor annotation packages for high throughput SNP platforms are under development. Column headers for the annotation that is currently available for each SNP is here:

```
> colnames(fData(annSnpset))
```

We store SNP-level attributes in the featureData slot. The command

```
> featureData(obj) <- getSnpAnnotation(annSnpset)
```

automatically loads the appropriate annotation package according to the annotation slot. The Bioconductor annotation packages must first be downloaded.
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)
```

For more detailed annotation on specific SNPs, see the R package RSNPper available at Bioconductor.

3 High throughput SNP classes

All that is needed to create an instance of `AnnotatedSnpCallSet` or `AnnotatedSnpCopyNumberSet`is a matrix of genotype calls and copy number, respectively, and their corresponding confidence scores. If estimates of both copy number and genotype calls are available, we can create an `AnnotatedSnpSet` that inherits methods from both `AnnotatedSnpCopyNumberSet` and `AnnotatedSnpCallSet`. In this way, *SNPchip* is completely independent of the pre-processing method used to produce probe-level summaries. To illustrate, the following code chunk loads a list of matrices obtained from normal subjects in the Hapmap project and pre-processed by CRLMM (B. Carvalho *et. al*, *Biostatistics*, in press). Only every 10th SNP from the Xba 50k chip is included in the matrices.

```
> data(hapmap)
> str(hapmap)
```

```
List of 3
 $ calls          : int [1:5896, 1:5] 2 2 2 3 3 3 1 1 3 3 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5896] "SNP_A-1507972" "SNP_A-1641761" "SNP_A-1641781" "SNP_A-1641805" ...
  .. ..$ : chr [1:5] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
 $ callsConfidence: num [1:5896, 1:5] 566 326 202 668 674 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5896] "SNP_A-1507972" "SNP_A-1641761" "SNP_A-1641781" "SNP_A-1641805" ...
  .. ..$ : chr [1:5] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
 $ copyNumber     : num [1:5896, 1:5] 2.67 1.77 2.18 2.08 2.02 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5896] "SNP_A-1507972" "SNP_A-1641761" "SNP_A-1641781" "SNP_A-1641805" ...
  .. ..$ : chr [1:5] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
```

Each matrix in the list contains probeset summaries (rows) by column (samples). Currently, we only provide annotation for the Affymetrix SNP chips and so the rownames of the matrices should be Affymetrix probeset id's. For purposes of visualization, an identifier from any technology could be used so long as the SNP-level annotation stored in the featureData slot is a data.frame with columns `Physical.Position` and `chromosome`.

```
> rownames(hapmap$calls)[1:5]
```

```
[1] "SNP_A-1507972" "SNP_A-1641761" "SNP_A-1641781" "SNP_A-1641805"
[5] "SNP_A-1641825"
```

```
> rowSds <- function(x) apply(x, 1, "sd")
> colSds <- function(x) apply(x, 2, "sd")
> nr <- nrow(hapmap$copyNumber)
```

```
> nc <- ncol(hapmap$copyNumber)
> snpset <- new("AnnotatedSnpSet", calls = hapmap$calls,
+     callsConfidence = hapmap$callsConfidence, copyNumber = hapmap$copyNumber,
+     cnConfidence = matrix(NA, nr, nc), annotation = "pd.mapping50k.xba240",
+     chromosomeAnnotation = chromosomeAnnotation)

> library("pd.mapping50k.xba240")
> annSnpset <- getSnpAnnotation(snpset)
```

This may take several minutes depending on your internet connection. To do this manually using NetAffx annotation files, the annotation files can be downloaded from
`http://biostat.jhsph.edu/ iruczins/publications/sm/2006.scharpf.bioinfo/mapping/`. This object should be converted to an object of class `AnnotatedDataFrame` with SNPs in the same order as in the `AnnotatedSnpSet` object. To download a static data.frame of the NetAffx annotation for the 50k Xba SNP chip, execute the following command:

```
> try(load(url(paste(path, "/mapping/mapping50kXba240.rda",
+     sep = ""))))
```

`data.frame`s for the 50k Hind and the 250k Nsp and Sty chips are also available:

```
> try(load(url(paste(path, "/mapping/mapping50kHind240.rda",
+     sep = ""))))
> try(load(url(paste(path, "/mapping/mapping250kNsp.rda",
+     sep = ""))))
> try(load(url(paste(path, "/mapping/mapping250kSty.rda",
+     sep = ""))))
```

Below, we illustrate how one might convert output from Affymetrix CNAT software to an object of class `AnnotatedSnpSet`. For instance, any one of the .txt files for the CEPH trios provided at the Affymetrix website can be converted as follows

```
> cnat <- read.table("100k_trios.Hind.1.txt", as.is = TRUE,
+     sep = "\t", header = TRUE, row.names = 1, skip = 0)
> cn <- as.matrix(cnat[, grep("SPA_CN", colnames(x))])
> calls <- cnat[, grep("_Call", colnames(x))]
> calls[calls == "AA"] <- 1
> calls[calls == "AB"] <- 2
> calls[calls == "BB"] <- 3
> calls[calls == "NoCall"] <- 4
> calls <- matrix(as.integer(as.matrix(calls)), nc = dim(calls)[2],
+     byrow = FALSE)
> cnConfidence <- as.matrix(cnat[, grep("SPA_pVal", colnames(cnat))])
> callsConfidence <- as.matrix(cnat[, grep("LOH", colnames(cnat))])
> rownames(calls) <- rownames(cn) <- x$Probe.Set
> rownames(cnConfidence) <- rownames(callsConfidence) <- x$Probe.Set
> colnames(cn) <- colnames(calls) <- substr(colnames(cn),
+     1, 7)
> colnames(callsConfidence) <- colnames(cnConfidence) <- substr(colnames(cn),
+     1, 7)
> trios <- new("AnnotatedSnpSet", calls = calls, copyNumber = copyNumber,
+     callsConfidence = callsConfidence, cnConfidence = cnConfidence,
+     annotation = "pd.mapping50k.hind240", chromosomeAnnotation = chromosomeAnnotation)
```

```
> library("pd.mapping50k.hind240")
> featureData(trios) <- getSnpAnnotation(trios)
```

The SNP-level annotation for the trios data can be retrieved as described previously.

```
> cnset <- as(annSnpset, "AnnotatedSnpCopyNumberSet")
> plotSnp(object = cnset, chromosomes = 1:10, samples = 1:3,
+     cex = 5, pch = ".", width.right = 3, cex.axis = 1,
+     cex.legend = c(1.2, 1.2), legend = c(TRUE, FALSE))
```

4 Descriptive and statistical summaries

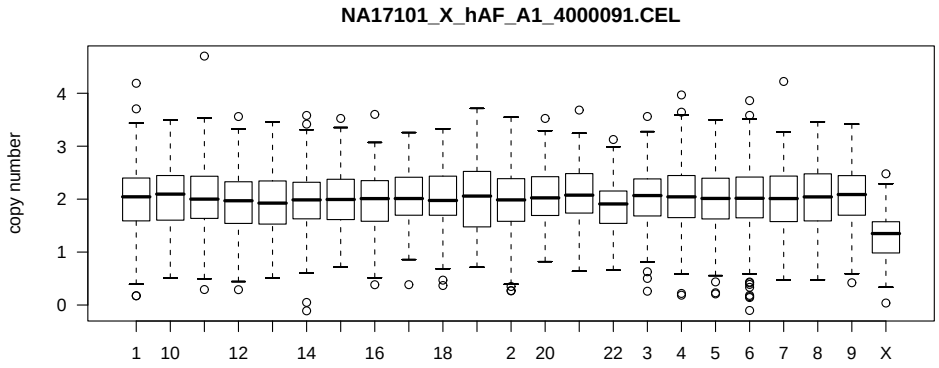
Descriptive statistics for copy number and genotype calls are provided with the `summary` method. For each chromosome in the `AnnotatedSnpSet`, `summary` calculates the average and standard deviation of the copy number estimates, as well as the % homozygous and heterozygous calls. In addition, summary calculates the average copy number, standard deviation, % homozygous and heterozygous across all autosomes in the `AnnotatedSnpSet`. 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 `AnnotatedSnpSet`.

```
> x <- summary(annSnpset, digits = 1)
> str(x)
```

```
List of 5
 $ avgCN   : num [1:6, 1:23] 2 2 2 2 2 2 2 2 2 2 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:6] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
  .. ..$ : chr [1:23] "1" "2" "3" "4" ...
 $ sdCN    : num [1:6, 1:23] 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:6] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
  .. ..$ : chr [1:23] "1" "2" "3" "4" ...
 $ %NoCalls: num [1:6, 1:23] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:6] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
  .. ..$ : chr [1:23] "1" "2" "3" "4" ...
 $ %Hom    : num [1:6, 1:23] 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 0.7 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:6] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
  .. ..$ : chr [1:23] "1" "2" "3" "4" ...
 $ %Het    : num [1:6, 1:23] 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.3 0.3 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:6] "NA17101_X_hAF_A1_4000091.CEL" "NA17102_X_hAF_A2_4000091.CEL" "NA17103_X_hAF_A3_4
  .. ..$ : chr [1:23] "1" "2" "3" "4" ...
```

Boxplot by chromosome:

```
> par(mfrow = c(1, 1), mar = c(4, 4, 3, 1), las = 1)
> boxplot(split(copyNumber(annSnpset[, 1]), chromosome(annSnpset)),
+     ylab = "copy number", main = sampleNames(annSnpset)[1])
```



Smoothing example

The basic unit for all of the above visualization tools and summary methods is an `AnnotatedSnpSet` of a single chromosome. For instance, `plotSnp` converts the `AnnotatedSnpSet` to a list of `AnnotatedSnpSet`, where each element in the list is an `AnnotatedSnpSet` of a single chromosome. In the code below, we are interested in a quick method for smoothing copy number estimates for each chromosome and apply a loess smoother to each chromosome. The following code chunk first assigns heterozygous calls to the integer 1 and homozygous calls to the integer zero. In this way, regions of deletions will have homozygous calls of zero. We simulated a deletion of 50 consecutive SNPs and then converted the `AnnotatedSnpSet` to a list where each element in the list is an `AnnotatedSnpSet` for a particular chromosome.

```
> sim1 <- annSnpset[chromosome(annSnpset) %in% 1:5, 1:3]
> sim1

AnnotatedSnpSet (storageMode: lockedEnvironment)
assayData: 2212 features, 3 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_A2_4000091
  .CEL, NA17103_X_hAF_A3_4000091.CEL
  varLabels and varMetadata description: none
featureData
An object of class "AnnotatedDataFrame"
  rowNames: SNP_A-1507972, SNP_A-1641781, ..., SNP_A-1759046  (2212 total)
  varLabels and varMetadata description:
    dbsnp_rs_id: dbsnp_rs_id
    chrom: chrom
    ...: ...
    enzyme: enzyme
    (8 total)
Annotation [1] "pd.mapping50k.xba240"

chromosomeAnnotation
    centromereStart centromereEnd chromosomeSize
```

```
1         121147476     123387476      245522847
2          91748045      94748045      243018229
...             ...           ...            ...
Y          11237315      12237315       57701691
```

```
> tmp <- sim1[chromosome(sim1) == "1", ]
> tmp <- tmp[order(position(tmp)), ]
> snps <- featureNames(tmp)[101:150]
> ids <- match(snps, featureNames(sim1))
> copyNumber(sim1)[ids, 1] <- copyNumber(sim1)[ids, 1] -
+     1
> calls(sim1)[ids, 1] <- 1

> sim2 <- sim1
> calls(sim2)[calls(sim2) == 1 | calls(sim2) == 3] <- 0
> calls(sim2)[calls(sim2) == 2] <- 1
> sim.list <- split(sim2, chromosome(sim2))
> sim.list[[1]]

AnnotatedSnpSet (storageMode: lockedEnvironment)
assayData: 469 features, 3 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_A2_4000091
  .CEL, NA17103_X_hAF_A3_4000091.CEL
  varLabels and varMetadata description: none
featureData
An object of class "AnnotatedDataFrame"
  rowNames: SNP_A-1642387, SNP_A-1643189, ..., SNP_A-1759036  (469 total)
  varLabels and varMetadata description:
    dbsnp_rs_id: dbsnp_rs_id
    chrom: chrom
    ...: ...
    enzyme: enzyme
    (8 total)
Annotation [1] "pd.mapping50k.xba240"

chromosomeAnnotation
    centromereStart centromereEnd chromosomeSize
1         121147476     123387476      245522847
2          91748045      94748045      243018229
...             ...           ...            ...
Y          11237315      12237315       57701691
```

```
> smoothChromosome <- function(obj, span) {
+     loessX <- function(X, location, span) {
+         fit <- loess(X ~ location, span = span)$fitted
+         return(fit)
+     }
```

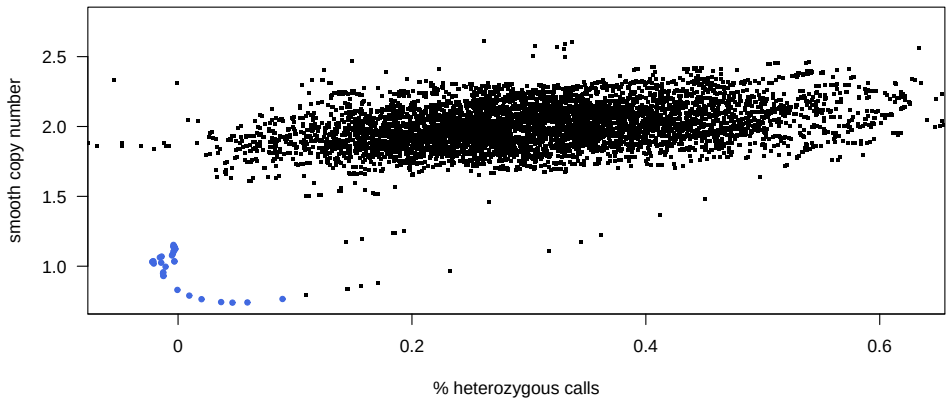
```
+     obj <- obj[order(position(obj)), ]
+     cn.smooth <- apply(copyNumber(obj), 2, loessX, position(obj),
+         span = span)
+     rownames(cn.smooth) <- featureNames(obj)
+     call.smooth <- apply(calls(obj), 2, loessX, location = position(obj),
+         span = span)
+     rownames(call.smooth) <- featureNames(obj)
+     copyNumber(obj) <- cn.smooth
+     calls(obj) <- call.smooth
+     obj
+ }
> smoothList <- lapply(sim.list, smoothChromosome, span = 1/10)
> smoothSet <- unsplitS4(smoothList, featureData(sim2))
```

Or equivalently,

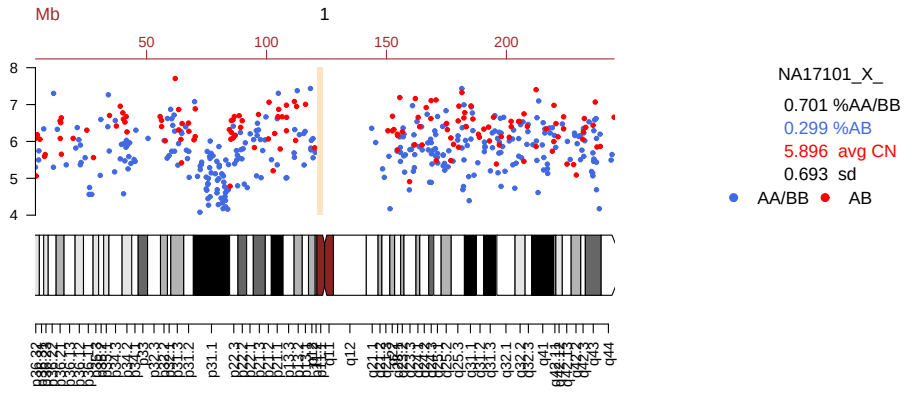
```
> smoothSet2 <- smoothSnp(sim1, 1:5, 1:3, span = 1/10)
> identical(copyNumber(smoothSet2), copyNumber(smoothSet))

[1] TRUE

> identical(calls(smoothSet2), calls(smoothSet))

[1] TRUE
```

A plot of the smoothed calls versus copynumber can be used to visualize the deletion and deciding on a threshold for calling deletions.

```
> par(las = 1, mar = c(5, 4, 0.5, 0.5), oma = rep(0, 4))
> plot(calls(smoothSet2), copyNumber(smoothSet2), ylim = range(copyNumber(smoothSet2)),
+     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(smoothSet2)), labels = pretty(calls(smoothSet2)))
> highlight <- calls(smoothSet2) <= 0.1 & copyNumber(smoothSet2) <=
+     1.5
> points(calls(smoothSet2)[highlight], copyNumber(smoothSet2)[highlight],
+     pch = 20, col = "royalblue", bg = "white")
```



```
> copyNumber(sim1) <- copyNumber(sim1) + 2
> plotSnp(sim1, 1, 1, log = "", width.right = 0.5, cex.legend = c(1.2,
+     1.2), legend.panel = c(TRUE, FALSE), cex = c(1,
+     1, 1), pch = rep(20, 3), legend.location = c("topright",
+     "bottomright"), addCytoband = TRUE)
```



5 Integration with other Bioconductor packages

To retreive additional annotation on the known SNP's in the region of this simulated deletion, we could use the *RSNPper*. The installation instructions for *RSNPper* is available at Bioconductor.

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

6 Session Information

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

- R version 2.6.0 Under development (unstable) (2007-07-08 r42150), `powerpc-apple-darwin8.10.0`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, splines, stats, tools, utils
- Other packages: Biobase 1.15.19, BufferedMatrix 1.1.3, BufferedMatrixMethods 1.1.4, DBI 0.2-3, RSQLite 0.5-4, SNPchip 1.1.12, affyio 1.5.0, oligo 1.1.5, preprocessCore 0.99.8