

VanillaICE: Hidden Markov Models for the Assessment of Chromosomal Alterations using High-throughput SNP Arrays

Robert Scharpf

March 10, 2008

1 Introduction

Chromosomal DNA is characterized by variation between individuals at the level of entire chromosomes (e.g. aneuploidy in which the chromosome copy number is altered), segmental changes (including insertions, deletions, inversions, and translocations), and changes to small genomic regions (including single nucleotide polymorphisms). A variety of alterations that occur in chromosomal DNA, many of which can be detected using high density single nucleotide polymorphism (SNP) microarrays, are linked to normal variation as well as disease and therefore of particular interest. These include changes in copy number (deletions and duplications) and genotype (e.g. the occurrence of regions of homozygosity). Hidden Markov models (HMM) are particularly useful for detecting such abnormalities, modeling the spatial dependence between neighboring SNPs. Here, we extend previous approaches that utilize HMM frameworks for inference in high throughput SNP arrays by integrating copy number, genotype calls, and the corresponding measures of uncertainty when available. Using simulated and real data, we demonstrate how confidence scores control smoothing in a probabilistic framework. The goal of this vignette is to provide a simple interface for fitting HMMs and plotting functions to help visualize the predicted states alongside the experimental data.

2 Simple Usage

2.1 Vanilla HMM

```
> library(VanillaICE)
```

Before fitting the HMM, the data must be organized into one of the following classes for high-throughput SNP data:

- `SnpCallSet` (genotype calls)
- `SnpCopyNumberSet` (copy number estimates)
- `oligoSnpSet` (genotype and copy number estimates)

When pre-processing Affymetrix SNP chips with the R package *oligo*, an object of one of the above classes is created and can be used directly with the HMMs described in this vignette. These classes can also be created from Illumina data as described in the *IlluminaHowTo* vignette. The simulated data provided with this package is an instance of class `oligoSnpSet`

```
> data(chromosome1)
> annotation(chromosome1)
```

```
[1] "pd.mapping50k.hind240,pd.mapping50k.xba240"

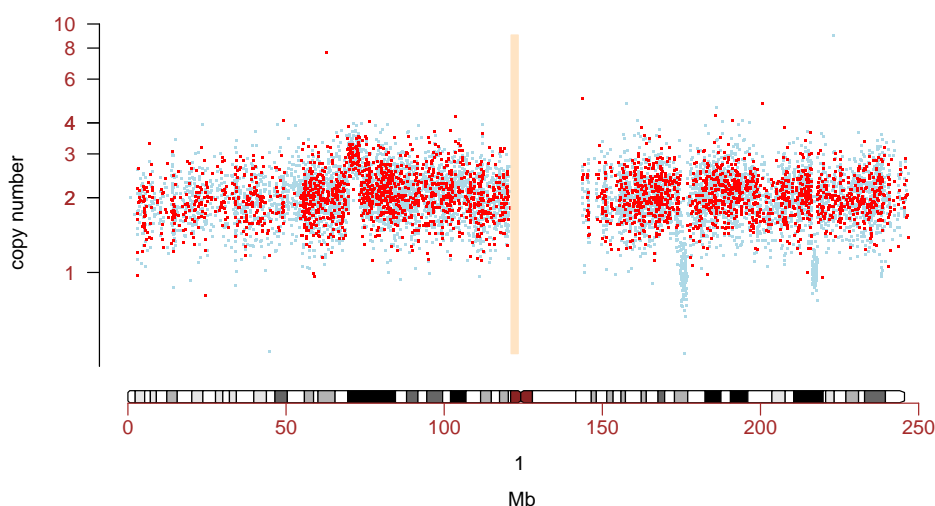
> chromosome1

oligoSnpSet (storageMode: lockedEnvironment)
assayData: 9165 features, 1 samples
  element names: calls, callsConfidence, cnConfidence, copyNumber
experimentData: use 'experimentData(object)'
Annotation: pd.mapping50k.hind240,pd.mapping50k.xba240
phenoData
An object of class "AnnotatedDataFrame"
  sampleNames: NA06993
  varLabels and varMetadata description:
    family: trio variable
    upd: uniparental isodisomy indicator
featureData
An object of class "AnnotatedDataFrame"
  rowNames: SNP_A-1677174, SNP_A-1718890, ..., SNP_A-1677548 (9165 total)
  varLabels and varMetadata description:
    dbsnp_rs_id: dbsnp_rs_id
    chromosome: chrom
    ...: ...
    enzyme: enzyme
    (8 total)
Annotation [1] "pd.mapping50k.hind240,pd.mapping50k.xba240"
```

Before deciding whether to fit a HMM, plot the data:

```
> gp <- new("ParSnpSet")
> gp <- getPar(gp, chromosome1)
> plotSnp(object = gp, snpset = chromosome1)
```

NULL



The HMM assumes that the copy number estimates, conditional on the hidden state, are approximately Gaussian. A log transformation will be performed automatically provided that the copy number estimates are all positive. See the documentation pages in the R package *VanillaICE* for more information about the `chromosome1` example dataset. Parameters for fitting the HMM are obtained by generating an instance of the class `HmmParameter`. By default, the HMM assumes the hidden states are deletion (D), normal (N), LOH (L), and amplification (A). See the `HmmParameterClass` vignette for more information. An instance of the class is created by the method `new`:

```
> params <- new("HmmParameter", chromosome1, states = c("D",
+             "N", "L", "A"), cn.location = c(1, 2, 2, 3))

[1] "Transforming copy number to log2 scale."
[1] "Calculating emission probabilities for genotype calls.... "
[1] "Calculating emission probabilities for copy number estimates... "
```

```
> class(params)

[1] "HmmParameter"
attr(,"package")
[1] "VanillaICE"
```

We may then fit the HMM by

```
> fit <- hmm(params, chromosome1)

[1] "Transforming copy number to log2 scale."

> fit

HmmPredict (storageMode: lockedEnvironment)
assayData: 9165 features, 1 samples
  element names: predictions
phenoData
  sampleNames: NA06993
  varLabels and varMetadata description:
    family: trio variable
    upd: uniparental isodisomy indicator
featureData
  featureNames: SNP_A-1677174, SNP_A-1718890, ..., SNP_A-1677548 (9165 total)
  fvarLabels and fvarMetadata description:
    chromosome: chromosome
    arm: chromosomal arm
    position: physical position
experimentData: use 'experimentData(object)'
Annotation: pd.mapping50k.hind240,pd.mapping50k.xba240

SnpClass: oligoSnpSet
hidden states: D N L A
breakpoints:
'data.frame':      11 obs. of  9 variables:
 $ id   : chr  "NA06993" "NA06993" "NA06993" "NA06993" ...
 $ chr  : chr  "1" "1" "1" "1" ...
 $ state: chr  "N" "L" "N" "A" ...
```

```

$ size : num  48.71  4.81 15.34  3.30 101.64 ...
$ N      : num  997 102 902 202 3797 ...
$ start: num  0.837 49.598 54.499 69.854 73.174 ...
$ last : num  49.5 54.4 69.8 73.2 174.8 ...
$ prev : num    NA 49.5 54.4 69.8 73.2 ...
$ next : num  49.6 54.4 69.9 73.2 174.8 ...

```

```
> breakpoints(fit)
```

	id	chr	state	size	N	start	last
1.1	NA06993	1	N	48.708312	997	0.836727	49.54504
1.2	NA06993	1	L	4.811945	102	49.597810	54.40975
1.3	NA06993	1	N	15.339118	902	54.498950	69.83807
1.4	NA06993	1	A	3.299434	202	69.854466	73.15390
1.5	NA06993	1	N	101.640222	3797	73.174070	174.81429
1.6	NA06993	1	D	1.985684	100	174.815096	176.80078
1.7	NA06993	1	N	39.439073	1900	176.800844	216.23992
1.8	NA06993	1	D	1.586808	99	216.286002	217.87281
1.9	NA06993	1	N	20.410491	901	217.892177	238.30267
1.10	NA06993	1	D	0.097921	5	238.319943	238.41786
1.11	NA06993	1	N	8.431130	160	238.429864	246.86099

	prev	next
1.1	NA	49.59781
1.2	49.54504	54.40975
1.3	54.40975	69.85447
1.4	69.83807	73.17407
1.5	73.15390	174.81429
1.6	174.81429	176.80084
1.7	176.80078	216.28600
1.8	216.23992	217.89218
1.9	217.87281	238.31994
1.10	238.30267	238.41786
1.11	238.41786	246.86099

FIXME: prev and next are NA in the breakpoints matrix.

Summary statistics for the breakpoints (more useful when multiple chromosomes and samples are in the fit object)

```
> summary(fit)
```

\$L

	Freq	med(length)	avg(length)	sd(length)	med(n.snp)	avg(n.snp)
1	1	4811.945	4811.945	NA	102	102

sd(n.snp)

1	NA
---	----

\$A

	Freq	med(length)	avg(length)	sd(length)	med(n.snp)	avg(n.snp)
1	1	3299.434	3299.434	NA	202	202

sd(n.snp)

1	NA
---	----

```

$D
  Freq med(length) avg(length) sd(length) med(n.snp) avg(n.snp)
1    3   1586.808   1223.471   994.949         99         68
sd(n.snp)
1    54.562

```

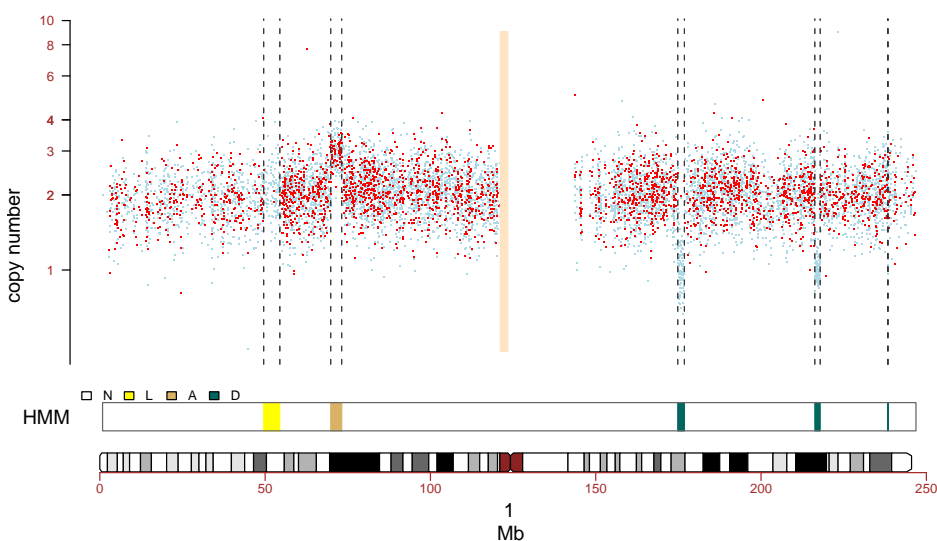
See [2] for a more complete description of the simulated dataset and the features detected by this HMM. We may plot the data along with the predictions as follows:

```

> fn <- featureNames(params)
> idx <- match(fn, featureNames(chromosome1))
> chromosome1 <- chromosome1[idx, ]
> gp <- new("ParHmmSnpSet")
> gp <- getPar(gp, snpset = chromosome1)
> gp$abline.v <- TRUE
> plotSnp(object = gp, snpset = chromosome1, hmmPredict = fit)

```

NULL



2.2 Integrating Confidence Estimates (ICE)

In this section, we illustrate how one may fit an HMM that incorporates confidence estimates of the SNP-level summaries for genotype calls and copy number. Confidence scores (inverse of standard errors) are available for this object (see Section 2.3 for how confidence scores were derived). This information is incorporated into the HMM emission probabilities in the following way:

```

> params.ice <- new("HmmParameter", chromosome1, states = c("D",
+   "N", "L", "A"), cn.ICE = TRUE)

[1] "Transforming copy number to log2 scale."
[1] "Calculating emission probabilities for genotype calls.... "
[1] "Calculating emission probabilities for copy number estimates... "

```

We may also incorporate the confidence scores for the genotype calls:

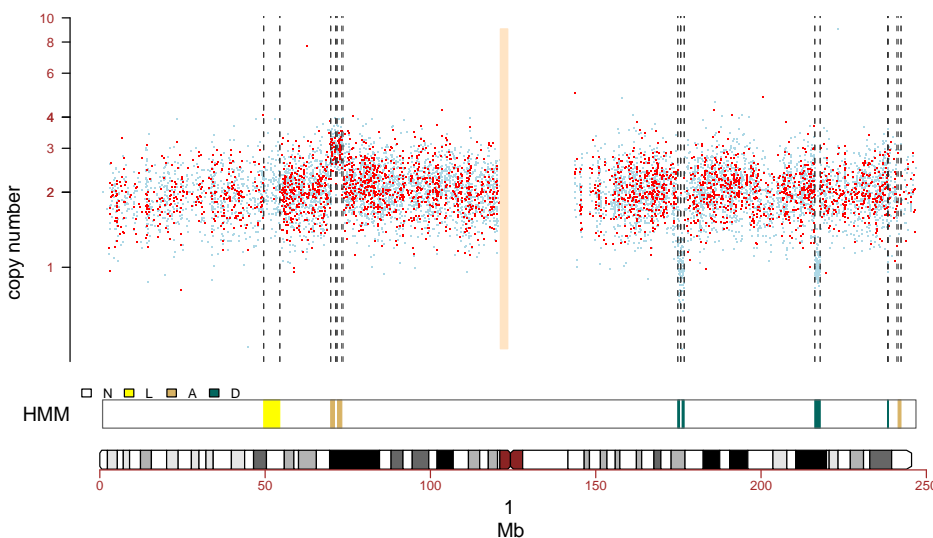
```
> params.ice <- new("HmmParameter", chromosome1, states = c("D",
+ "N", "L", "A"), cn.ICE = TRUE, gt.ICE = TRUE)
```

We fit the HMM in the usual way

```
> fit.ice <- hmm(params.ice, chromosome1)
[1] "Transforming copy number to log2 scale."
```

and plot the results using the same graphical parameters as above:

```
> plotSnp(object = gp, snpset = chromosome1, hmmPredict = fit.ice)
NULL
```



Note that the ICE HMM correctly identifies the simulated normal segments in features B and C (the normal segments were simulated to have high confidence scores). Additionally, the ICE HMM detects the micro-amplification in region E (also simulated to have high confidence scores).

2.3 Confidence scores

Confidence scores for genotype calls We suggest using the CRLMM algorithm [1] for genotype calls. CRLMM (in the R package *oligo*) provides confidence scores ($S_{\widehat{GT}}$) of the genotype estimates (\widehat{GT}). From 269 HapMap samples assayed on the Affymetrix 50k Xba and Hind chips, we have a gold standard of the true genotype defined by the consensus of the HapMap centers. We use kernel based density estimates to obtain

$$f \left\{ S_{\widehat{HOM}} \mid \widehat{HOM}, HOM \right\}, f \left\{ S_{\widehat{HOM}} \mid \widehat{HOM}, HET \right\}, f \left\{ S_{\widehat{HET}} \mid \widehat{HET}, HOM \right\}, \text{ and } f \left\{ S_{\widehat{HET}} \mid \widehat{HET}, HET \right\} \quad (1)$$

separately for the Xba and Hind 50k chips. The first term in (1), for example, denotes the density of the scores when the genotype is correctly called homozygous (\widehat{HOM}) and the true genotype is homozygous (HOM). See [2] for a more complete description of the methods. The data needed to estimate these densities is stored in the experiment data package *callsConfidence*. *callsConfidence* is available from the author's website.

Confidence scores for copy number estimates To illustrate how standard errors of the copy number estimate could be integrated in the HMM, the R object `chromosome1` contains standard errors simulated from a shifted Gamma: $\text{Gamma}(1, 2) + 0.3$, where 1 is the shape parameter and 2 is the rate parameter. To ascertain the effect of qualitatively high confidence scores on the ICE HMM, we scaled a robust estimate of the copy number standard deviation by $\frac{1}{2}$. Similarly, to simulate less precise $\widehat{\text{CN}}$ we scaled ϵ by 2. For more detailed information about how the data in the `chromosome1` was generated, see the documentation for this object in the R package *VanillaICE*.

3 The HmmParameter class

The minimal information required to create an instance of class `HmmParameter` is

1. an object inheriting from the class `SnplLevelSet`. For instance, an `oligoSnplSet`.
2. a vector of names for the hidden states. Only the name for the normal state, “N”, requires a controlled vocabulary.
3. optionally, one may define any of the following arguments when creating an instance of the class:
 - `SCALE`
 - `tau.scale`
 - `tau`
 - `pi`
 - `beta`
 - `cn.location`
 - `cn.scale`
 - `cn.ICE`
 - `gt.ICE`
 - `gt.gte`
 - `gt.state`
 - `gt.confidence`
 - `gt.confidence.states`
 - `notes`
 - `annotation`

An instance of the class is created by the method `new`:

```
> params <- new("HmmParameter", chromosome1, states = c("D",
+ "N", "L", "A"), cn.location = log2(c(1, 2, 2, 3)))
```

The object `params` contains all of the parameters needed for fitting the HMM, including the transition probabilities (`tau`), emission probabilities (`beta`), and initial state probabilities (`pi`). A summary of the parameters contained in the object `params` is obtained by

```
> params
```

```

Formal class 'HmmParameter' [package "VanillaICE"] with 7 slots
..@ hmmOptions :Formal class 'HmmOptions' [package "VanillaICE"] with 11 slots
.. .. ..@ SnpClass          : atomic [1:1] oligoSnpSet
.. .. ..- attr(*, "package")= chr "oligoClasses"
.. .. ..@ states            : chr [1:4] "D" "N" "L" "A"
.. .. ..@ cn.location       : num [1:4] 1 2 2 3
.. .. ..@ cn.robustSE       : Named num 0.31
.. .. ..- attr(*, "names")= chr "NA06993"
.. .. ..@ cn.ICE            : logi FALSE
.. .. ..@ gt.ICE            : logi FALSE
.. .. ..@ gt.gte            : chr "not used if gt.ICE=FALSE"
.. .. ..@ gte.state         : Named num [1:4] 0.999 0.75 0.999 0.75
.. .. ..- attr(*, "names")= chr [1:4] "D" "N" "L" "A"
.. .. ..@ gt.confidence.states: chr(0)
.. .. ..@ annotation        : chr "pd.mapping50k.hind240,pd.mapping50k.xba240"
.. .. ..@ notes             : chr ", gt.state is P(gte = HOM | state)"
..@ tau                : num [1:9164] 0.973 0.986 1.000 1.000 1.000 ...
..@ tau.scale          : num [1:4, 1:4] 1 1 0.75 0.75 1.5 1 1.5 1.5 0.75 1 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:4] "D" "N" "L" "A"
.. .. ..$ : chr [1:4] "D" "N" "L" "A"
..@ pi                 : Named num [1:4] -8.006 -0.001 -8.006 -8.006
.. ..- attr(*, "names")= chr [1:4] "D" "N" "L" "A"
..@ beta               : num [1:9165, 1:4, 1] -4.9494 -2.8644 -7.0156 0.0893 -1.0799 ...
..@ featureData:Formal class 'AnnotatedDataFrame' [package "Biobase"] with 4 slots
.. .. ..@ varMetadata      :'data.frame':      3 obs. of  1 variable:
.. .. .. ..$ labelDescription: chr [1:3] "chromosome" "chromosomal arm" "physical position"
.. .. ..@ data              :'data.frame':      9165 obs. of  3 variables:
.. .. .. ..$ chromosome: chr [1:9165] "1" "1" "1" "1" ...
.. .. .. ..$ arm       : chr [1:9165] "p" "p" "p" "p" ...
.. .. .. ..$ position  : int [1:9165] 836727 2224111 2915399 2926730 2941104 2941694 2963436 3084986
.. .. ..@ dimLabels        : chr [1:2] "rowNames" "columnNames"
.. .. ..@ __classVersion__:Formal class 'Versions' [package "Biobase"] with 1 slots
.. .. .. ..@ .Data:List of 1
.. .. .. .. ..$ : int [1:3] 1 1 0
..@ notes               : chr(0)

```

Emission probabilities. The emission probabilities are stored as an array in the `params` object. The emission probability array has dimension $R \times S \times C$, where S is the number of hidden states, R is the number of rows (SNPs), and C is the number of samples. To obtain the emission probabilities for the i th SNP, one may use `[` to subset. For instance, the emission probabilities for the 5th SNP are:

```

> tmp <- params[5, 1]
> exp(Beta(tmp))
, , 1

```

```

      [,1]      [,2]      [,3]      [,4]
[1,] 0.3396269 0.2710860 0.3610865 0.002263227

```

Transition probabilities. The probability of remaining in the same state, $P(S_t = S_{t+1})$ (the diagonal of the transition probability matrix) is a function of the distance (d) between SNPs: $e^{-2d(100 \times 1e6)}$.

This value is stored in the slot `tau` of the `params` object. The probability of leaving a state is ϵ , where $\epsilon = 1 - P(S_t = S_{t+1})$. The ϵ is split among $S - 1$ states. By default, the probability of transitioning from an altered state back to the normal state is twice as likely as the probability of transitioning between two altered states. The weights for ϵ are provided in the `tau.scale` matrix in the R object `params`

```
> params@tau.scale

      D   N   L   A
D 1.00 1.5 0.75 0.75
N 1.00 1.0 1.00 1.00
L 0.75 1.5 1.00 0.75
A 0.75 1.5 0.75 1.00
```

and can be adjusted by the `SCALE` argument when creating an instance of the class. For illustration, one could make the probability of transitioning from an altered state to a normal state 10 times as likely as the probability of transitioning between two altered states by the following command:

```
> params <- new("HmmParameter", chromosome1, states = c("D",
+ "N", "L", "A"), cn.location = c(1, 2, 2, 3), SCALE = 10)

[1] "Transforming copy number to log2 scale."
[1] "Calculating emission probabilities for genotype calls.... "
[1] "Calculating emission probabilities for copy number estimates... "
```

To retrieve the copy number and genotype for the 5th SNP, one should use `match`:

```
> i <- match(names(tmp), featureNames(chromosome1))
> c(copyNumber(chromosome1[i, ]), calls(chromosome1[i,
+ ]))

numeric(0)
```

4 The HmmPredict Class

The output from the HMM is an instance of the `HmmPredict` class and contains the predicted states as well as the breakpoints for the different states.

```
> fit

HmmPredict (storageMode: lockedEnvironment)
assayData: 9165 features, 1 samples
  element names: predictions
phenoData
  sampleNames: NA06993
  varLabels and varMetadata description:
    family: trio variable
    upd: uniparental isodisomy indicator
featureData
  featureNames: SNP_A-1677174, SNP_A-1718890, ..., SNP_A-1677548 (9165 total)
  fvarLabels and fvarMetadata description:
    chromosome: chromosome
    arm: chromosomal arm
    position: physical position
```

```
experimentData: use 'experimentData(object)'
Annotation: pd.mapping50k.hind240,pd.mapping50k.xba240
```

```
  SnpClass: oligoSnpSet
hidden states:  D N L A
breakpoints:
'data.frame':      11 obs. of  9 variables:
 $ id   : chr  "NA06993" "NA06993" "NA06993" "NA06993" ...
 $ chr  : chr  "1" "1" "1" "1" ...
 $ state: chr  "N" "L" "N" "A" ...
 $ size : num  48.71  4.81 15.34  3.30 101.64 ...
 $ N    : num  997 102 902 202 3797 ...
 $ start: num  0.837 49.598 54.499 69.854 73.174 ...
 $ last : num  49.5 54.4 69.8 73.2 174.8 ...
 $ prev : num    NA 49.5 54.4 69.8 73.2 ...
 $ next : num  49.6 54.4 69.9 73.2 174.8 ...
```

The breakpoints are provided as a `data.frame`:

```
> breaks <- breakpoints(fit)
> breaks <- breaks[breaks[, "state"] != "N", ]
```

One may order the altered states from biggest to smallest for each chromosome as follows:

```
> breaks[order(breaks[, "chr"], breaks[, "size"], decreasing = TRUE),
+ ]
```

	id	chr	state	size	N	start	last	prev
1.2	NA06993	1	L	4.811945	102	49.59781	54.40975	49.54504
1.4	NA06993	1	A	3.299434	202	69.85447	73.15390	69.83807
1.6	NA06993	1	D	1.985684	100	174.81510	176.80078	174.81429
1.8	NA06993	1	D	1.586808	99	216.28600	217.87281	216.23992
1.10	NA06993	1	D	0.097921	5	238.31994	238.41786	238.30267
	next							
1.2				54.40975				
1.4				73.17407				
1.6				176.80084				
1.8				217.89218				
1.10				238.41786				

The `summary` method returns a list where each element in the list provides statistics for an altered states. For each chromosome, the mean, median, and standard deviation of the size of the features and number of SNPs involved are reported. For instance, in this HMM the altered states were loss of heterozygosity (L), amplification of copy number (A), and deletion of copy number (D). Because we only have one sample and one chromosome in the `fit` example, the `summary` method is not that useful:

```
> summary(fit)

$L
  Freq med(length) avg(length) sd(length) med(n.snp) avg(n.snp)
1    1    4811.945    4811.945         NA         102         102
  sd(n.snp)
1         NA
```

```

$A
  Freq med(length) avg(length) sd(length) med(n.snp) avg(n.snp)
1    1    3299.434    3299.434         NA         202         202
  sd(n.snp)
1         NA

$D
  Freq med(length) avg(length) sd(length) med(n.snp) avg(n.snp)
1    3    1586.808    1223.471    994.949         99         68
  sd(n.snp)
1    54.562

```

5 HMMs for different classes of data

5.1 Copy number

The method `hmm` has a different set of underlying hidden states depending on whether copy number estimates, genotype calls, or both are available. When only copy number estimates are available, the hidden states (for autosomes) are hemizygous or homozygous deletion (one or fewer copies), normal (two copies), and amplification (three or more copies). The corresponding data class is `SnpcopyNumberSet`. To illustrate, we convert the `chromosome1` example to an object of this class and fit the HMM.

```

> chr1.cn <- as(chromosome1, "SnpcopyNumberSet")
> params.cn <- new("HmmParameter", snpset = chr1.cn, cn.location = 1:3,
+   states = c("D", "N", "A"))

[1] "Transforming copy number to log2 scale."
[1] "Calculating emission probabilities for copy number estimates... "

> fit.cn <- hmm(params.cn, chr1.cn)

[1] "Transforming copy number to log2 scale."

> breakpoints(fit.cn)

```

	id	chr	state	size	N	start	last
1.1	NA06993	1	N	43.865725	877	0.836727	44.70245
1.2	NA06993	1	D	0.040126	2	44.722116	44.76224
1.3	NA06993	1	N	25.058717	1122	44.779351	69.83807
1.4	NA06993	1	A	3.299434	202	69.854466	73.15390
1.5	NA06993	1	N	101.640222	3797	73.174070	174.81429
1.6	NA06993	1	D	1.985748	101	174.815096	176.80084
1.7	NA06993	1	N	39.313361	1899	176.926556	216.23992
1.8	NA06993	1	D	1.586808	99	216.286002	217.87281
1.9	NA06993	1	N	20.410491	901	217.892177	238.30267
1.10	NA06993	1	D	0.097921	5	238.319943	238.41786
1.11	NA06993	1	N	8.431130	160	238.429864	246.86099

```

      prev      next
1.1      NA 44.70245
1.2 44.70245 44.76224
1.3 44.76224 69.85447

```

```

1.4 69.83807 73.17407
1.5 73.15390 174.81429
1.6 174.81429 176.80084
1.7 176.80084 216.28600
1.8 216.23992 217.89218
1.9 217.87281 238.31994
1.10 238.30267 238.41786
1.11 238.41786 246.86099

```

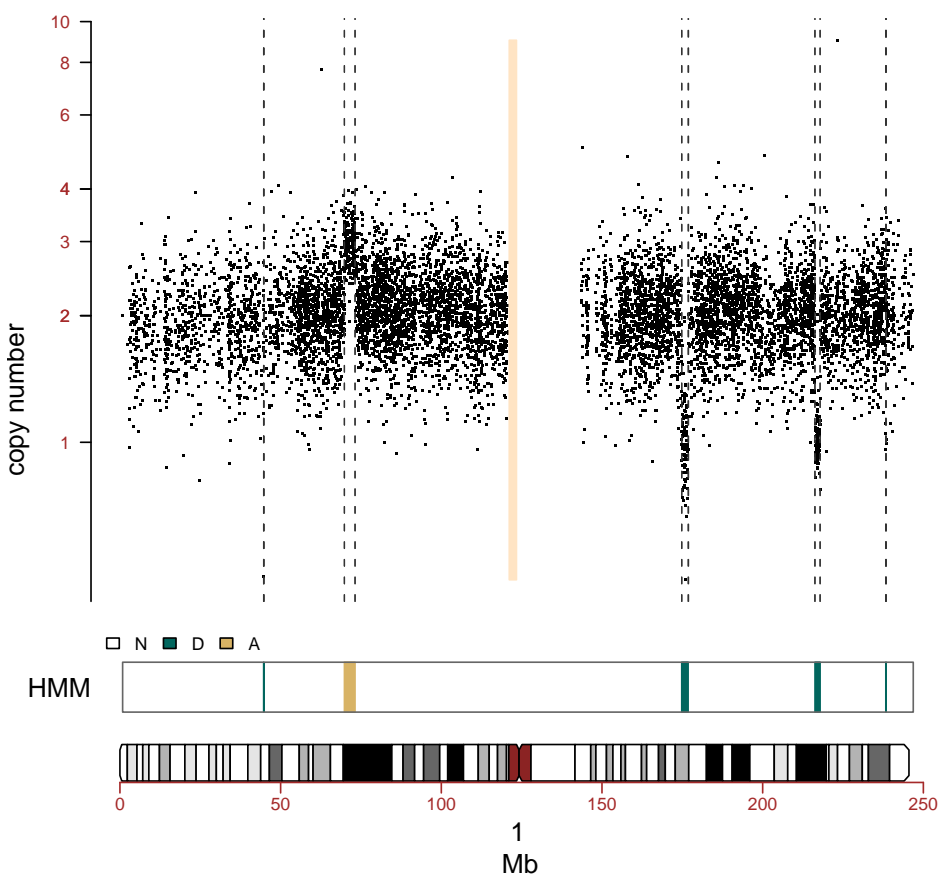
```

> graph.par <- new("ParHmmSnpCopyNumberSet")
> graph.par <- getPar(graph.par, chr1.cn)
> graph.par$abline.v <- TRUE

> plotSnp(graph.par, chr1.cn, fit.cn)

```

NULL



5.2 Genotype calls

When only genotype calls are available, the hidden states are loss and retention (ret) of heterozygosity. We define *loss* to be a sequence of homozygous SNPs longer than what we would expect to observe by

chance. Note that many long stretches of homozygosity may occur as a result of a population sharing a common underlying haplotype structure; loss predictions from an HMM fit to an individual do not necessarily reflect the 'loss' of an allele in that individual. For illustration, we convert the `chromosome1` example to an object of class `HmmSnpCallSet` and refit the HMM.

```
> chr1.calls <- as(chromosome1, "SnpCallSet")
> params.calls <- new("HmmParameter", snpset = chr1.calls,
+   states = c("L", "N"))

[1] "Calculating emission probabilities for genotype calls.... "
```

```
> fit.calls <- hmm(params.calls, chr1.calls)
> breakpoints(fit.calls)
```

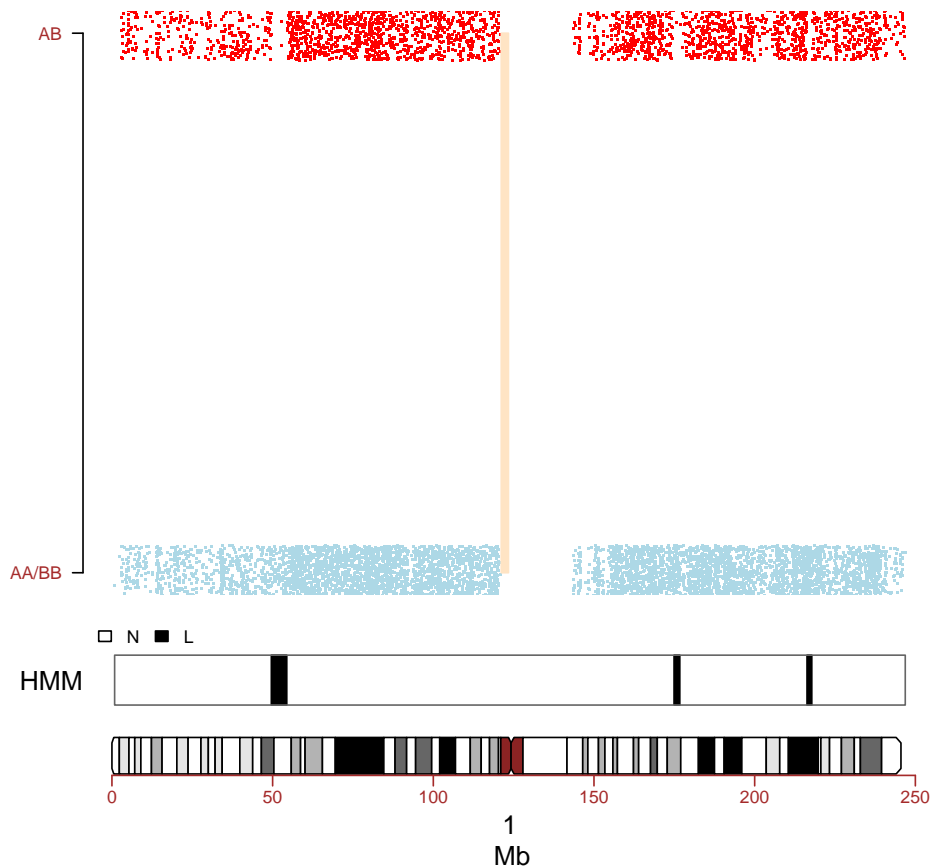
	id	chr	state	size	N	start	last	prev
1.1	NA06993	1	N	48.708312	997	0.836727	49.54504	NA
1.2	NA06993	1	L	4.811945	102	49.597810	54.40975	49.54504
1.3	NA06993	1	N	120.350649	4907	54.498950	174.84960	54.40975
1.4	NA06993	1	L	1.788366	92	174.915701	176.70407	174.84960
1.5	NA06993	1	N	39.439518	1902	176.800399	216.23992	176.70407
1.6	NA06993	1	L	1.485826	97	216.286002	217.77183	216.23992
1.7	NA06993	1	N	28.988981	1068	217.872013	246.86099	217.77183

```
      next
1.1  49.59781
1.2  54.40975
1.3 174.91570
1.4 176.80040
1.5 216.28600
1.6 217.87201
1.7 246.86099

> gp <- new("ParHmmSnpCallSet")
> graph.par <- getPar(gp, chr1.calls)

> plotSnp(graph.par, chr1.calls, fit.calls)

NULL
```



5.3 Genotype calls and copy number

Section 2 illustrates how one may fit the HMM to objects of class `oligoSnpSet`.

More documentation about the classes can be found in the documentation for the R package *VanillaICE*.

6 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, stats, tools, utils
- Other packages: Biobase 1.99.0, RColorBrewer 1.0-1, SNPchip 1.3.17, VanillaICE 1.1.15, oligoClasses 1.1.15

References

- [1] Benilton Carvalho, Henrik Bengtsson, Terence P Speed, and Rafael A Irizarry. Exploration, normalization, and genotype calls of high-density oligonucleotide SNP array data. *Biostatistics*, 8(2):485–499, Apr 2007.
- [2] Robert B Scharpf, Giovanni Parmigiani, Jonathan Pevsner, and Ingo Ruczinski. A hidden Markov model for joint estimation of genotype and copy number in high-throughput SNP chips. Technical Report Working Paper 136, Johns Hopkins University, February 2007.