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

Robert Scharpf

October 14, 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 Locus-level estimates of copy number

To do: describe how to obtain locus-level estimates of copy number.

2.2 Hidden Markov model to smooth locus-level estimates

```
> library(VanillaICE)

> 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"
```

A visualization of the data:

```
> gp <- plot(chromosome1)
> show(gp)
```

The HMM for total copy number assumes that the copy number estimates, conditional on the hidden state, are approximately Gaussian. Fitting a HMM requires the following components:

- the hidden states
- the emission probabilities
- transition probabilities

Hidden states For the hidden Markov model, it is important that the SNPs (within a chromosome) are ordered by physical position.

```
> ann <- fData(chromosome1)[, c("chromosome", "position")]
> ann[, "chromosome"] <- chromosome2integer(ann[, "chromosome"])
> chromosome1 <- chromosome1[order(ann[, "chromosome"],
+     ann[, "position"]), ]
```

Next we specify the hidden states and the corresponding number for each state.

```
> states <- c("homozygousDeletion", "hemizygousDeletion",
+     "normal", "LOH", "3copyAmp", "4copyAmp")
> mu <- c(0.05, 1, 2, 2, 3, 4)
```

SNP-specific estimates of the uncertainty of the total copy number will be available from methods in Section 2.1 shortly. Below, we simply obtain a robust estimate of the copy number standard deviation across SNPs and use this estimate for all of the SNPs. Because we the log-transformed copy number estimates are more nearly Gaussian, we calculate a robust estimate of the standard deviation on the log scale, and use this estimate for all of the SNPs. This section will be updated.

```
> library(genefilter)
> CT <- copyNumber(chromosome1)
> sample.sd <- matrix(rowSds(t(log2(CT))), nrow(CT), ncol(CT))
```

Emission probabilities Locus-level estimates of copy number, when suitably transformed, are assumed to be approximately Gaussian-distributed. We may obtain emission probabilities as follows:

```
> logCT <- array(log2(CT), dim = c(nrow(CT), ncol(CT),
+     length(states)))
> dimnames(logCT) <- list(rownames(CT), colnames(CT),
+     states)
> logMu <- aperm(array(log2(mu), dim = c(length(states),
+     ncol(CT), nrow(CT))))
> logSd <- aperm(array(sample.sd, dim = c(length(states),
+     ncol(CT), nrow(CT))))
> dimnames(logMu) <- dimnames(logSd) <- dimnames(logCT)
> k <- which(!is.na(as.vector(logCT)))
> emission.logCT <- dnorm(as.vector(logCT)[k], as.vector(logMu)[k],
+     as.vector(logSd)[k])
> emission.logCT <- array(emission.logCT, dim = dim(logCT))
> logemission.logCT <- log(emission.logCT)
```

Alternatively, one may use the function copynumberEmission (does nearly the same as above, but with a few additional checks).

```
> logemission.logCT2 <- copynumberEmission(copynumber = CT,
+     states = states, mu = mu, uncertainty = sample.sd,
+     takeLog = TRUE, verbose = TRUE)
> dimnames(logemission.logCT) <- dimnames(logemission.logCT2)
> identical(logemission.logCT, logemission.logCT2)
```

```
[1] TRUE
```

```
> rm(logemission.logCT2)
> gc()
```

```
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 576762 15.5     984024 26.3   741108 19.8
Vcells 721552  5.6    1407279 10.8  1390582 10.7
```

Adding a small positive value to the log emission probabilities prevents breaks in the predicted states due to extreme values. This step is less important if SNP-specific estimates of uncertainty are available.

```
> logemission.logCT[logemission.logCT < -10] <- -10
```

Uncertainty estimates for the genotype calls can also be integrated. The genotype calls must be represented as integers (1 = AA, 2 = AB, 3 = BB, 4 = missing). We recommend using CRLMM to genotype – CRLMM provides genotypes for all SNPs on the array (no missing values) and confidence estimates for the call that takes into consideration the signal to noise ratio of the sample as well as a SNP-specific estimate of the uncertainty. Di-allelic genotype calls from CRLMM are useful, though incorrect for CNV. In particular, di-allelic genotype calls can help pinpoint hemizygous deletions (genotype calls of AA or BB correspond to 'A' and 'B') and copy-neutral regions of loss of heterozygosity (LOH). For the HMM, one must specify the probability of a homozygous genotype call (`probHomCall`) for each of the hidden states. The goal, then, is to have genotype emission probabilities that are informative for distinguishing the normal state versus hemizygous deletions or LOH, and somewhat agnostic for distinguishing between homozygous deletions/amplifications from normal regions.

To illustrate this idea, consider two scenarios where the true state is amplification of copy number (e.g., 3 copies) over a regions spanning 10 SNPs. Scenario 1: 7 of the 10 di-allelic genotype calls are

homozygous. Scenario 2: all 10 of the di-allelic genotype calls are homozygous. For the hidden Markov model, we assume that the probability of a homozygous genotype call is 0.99 for normal, homozygous deletion and amplification. For hemizygous deletion and copy-neutral LOH, we specify a probability of 0.9999. Note the following:

- The emission probabilities for the di-allelic genotype calls in scenarios 1 and 2 will be the same for normal, homozygous deletion, and amplification. The copy number emission probabilities will discriminate between these states.
- Hemizygous deletion and copy-neutral LOH are penalized severely in scenario 1. This penalization reflects our belief that the probability of observing 3 genotype errors (AB must be a genotype error for these states) in a small region.
- In scenario 2, hemizgyous deletion and copy-neutral LOH have a higher emission probability for the genotypes. However, because the probability of observing homozygous genotype calls for amplification and normal states is also high, most of the information for discriminating between and amplification containing 10 homozygous genotype and a copy-neutral region of LOH will be driven by the emission probability for copy number.

In summary, specifiying high probabilities for observing a homozygous genotype call in normal and copy number alterered states (as indicated below) allows fairly long stretches of homozygous genotype calls to occur by chance in any of the states. In our experience, sequences of 70 - 100 homozygous genotypes are fairly common and likely represent normal regions of the genome with fairly uniform haplotype structure. If any of the genotype calls are missing and missingness is not independent of the underlying hidden state, one may specify the probability of a missing genotype calls for each hidden state (`probMissing`). By default, the HMM will assume that missing genotype calls are independent of the underlying hidden state.

```
> probs <- c(0.99, 0.9999, 0.99, 0.9999, 0.99, 0.99)
> probMissing <- c(0.999, rep(0.01, 5))
> names(probs) <- states
> GT <- calls(chromosome1)
> genotypeEmission <- function(genotypes, states, probHomCall,
+     probMissing, verbose = TRUE) {
+     if (!is.numeric(genotypes))
+         stop("genotypes must be integers (1=AA, 2=AB, 3=BB, 4=missing")
+     emissionForGenotypes <- function(probHomGenotype,
+         genotypes) {
+         isHom <- which(as.vector(genotypes) == 1 | as.vector(genotypes) ==
+             3)
+         isHet <- which(as.vector(genotypes) == 2)
+         isMissing <- which(as.vector(genotypes) == 4 |
+             is.na(as.vector(genotypes)))
+         emission.gt <- rep(NA, length(genotypes))
+         emission.gt[isHom] <- probHomGenotype
+         emission.gt[isHet] <- 1 - probHomGenotype
+         emission.gt[isMissing] <- NA
+         emission.gt
+     }
+     emission.gt <- array(NA, dim = c(nrow(GT), ncol(GT),
+         length(states)))
+     for (j in 1:ncol(GT)) {
+         emission.gt[, j, ] <- sapply(probs, emissionForGenotypes,
```

```
+             genotypes = GT[, j])
+         if (any(is.na(emission.gt[, j, 1]))) {
+             missing <- is.na(emission.gt[, j, 1])
+             if (!missing(probMissing)) {
+                 if (length(probMissing) != length(states))
+                   stop("probMissing must be a numeric vector equal to the number of states")
+                 emission.gt[missing, j, ] <- matrix(probMissing,
+                   sum(missing), length(states), byrow = TRUE)
+             }
+             else {
+                 if (verbose)
+                   message("Argument probMissing is not specified. Assume that missing genotype calls
+                 emission.gt[missing, j, ] <- 1
+             }
+         }
+     }
+     dimnames(emission.gt) <- list(rownames(genotypes),
+         colnames(genotypes), states)
+     return(suppressWarnings(log(emission.gt)))
+ }
> logemission.gt <- genotypeEmission(genotypes = calls(chromosome1),
+     states = states, probHomCall = probs)
```

Conditional on the hidden state, we assume that the copy number and genotype are independent. Therefore, the emission probabilities for an HMM that models the copy number and genotypes jointly are computed by adding the emission probabilities (log-scale) for copy number and genotype:

```
> logemission <- logemission.gt + logemission.logCT
```

Transition probabilities We transform the physical distance between adjacent loci to an estimate of the genomic distance.

```
> tau <- exp(-2 * diff(ann[, "position"])/(100 * 1e+06))
```

Note that the above transition probabilities can be scaled by specifying a matrix of dimension S x S, where S is the number of hidden states. For instance, we define loss of heterozgosity in a single sample as a sequence of homozygous genotypes that is longer than what one would expect to observe by chance. One way to control the size (and number) of LOH regions detected is to scale the probability of transitioning to and from this state. A transition scale matrix of 1's would not modify the probability of transitioning between states.

Example with the first SNP.

```
> epsilon <- 1 - tau[1]
> M <- matrix(epsilon/(length(states) - 1), length(states),
+     length(states))
> dimnames(M) <- list(states, states)
> diag(M) <- tau[1]
> all(rowSums(M) == 1)
```

```
[1] TRUE
```

Note that by default, the transition probability matrix distributes the epsilon equally to the remaining states. We may alter how the epsilon is distributed by scaling this matrix. For instance,

```
> tau.scale <- matrix(1, length(states), length(states))
> dimnames(tau.scale) <- list(states, states)
> tau.scale["normal", "LOH"] <- 1e-04
```

The other states must be rescaled (here, we rescale the other states by a factor `scale`)

```
> S <- length(states)
> scale <- (S - 1)/(S - 2 + 1e-04)
> tau.scale["normal", c("homozygousDeletion", "hemizygousDeletion",
+     "3copyAmp", "4copyAmp")] <- scale
> all(round(rowSums(M * tau.scale), 5) == 1)
```

```
[1] TRUE
```

Initial state probabilities

```
> initialStateProb <- rep(1e-04, length(states))
> initialStateProb[states == "normal"] <- 1 - (length(states) -
+     1) * 1e-04
```

Fitting the HMM We use the Viterbi algorithm to find the sequence of hidden states that maximizes the probability of the observed data.

```
> fit <- viterbi(initialStateProbs = log(initialStateProb),
+     emission = logemission[, 1, ], tau = tau)
> fit2 <- viterbi(initialStateProbs = log(initialStateProb),
+     emission = logemission[, 1, ], tau = tau, tau.scale = tau.scale)
```

Note that in the above example, scaling the transition probability matrix did not affect the predicted sequence of hidden states.

```
> table(fit)
```

```
fit
   2    3    5
 197 8764  204
```

```
> results <- findBreaks(x = fit, states = states, position = ann[,
+     "position"], chromosome = ann[, "chromosome"], sample = colnames(CT))
> results[results$state != "normal", ]
```

```
   sample chr     start       end  nbases nprobes             state
2 NA06993   1  69854466  73174389 3319923     204          3copyAmp
4 NA06993   1 174815096 176704067 1888971      98 hemizygousDeletion
6 NA06993   1 216286002 217872810 1586808      99 hemizygousDeletion
```

3 Using S4 classes/methods

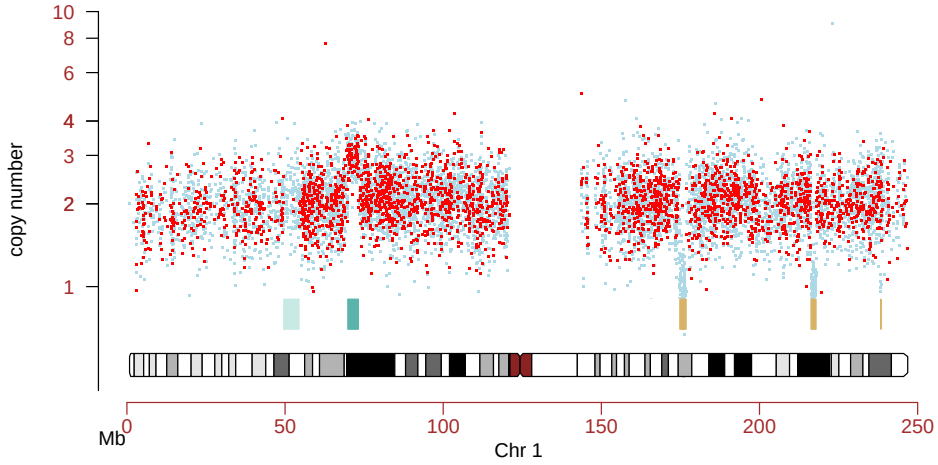
The objective of developing classes is to facilitate the process of fitting an HMM and to more effectively keep the assaydata and metadata together in one object. The following code uses classes to do pretty much the same as above. We begin by reproducing the `chromosome1` object, an object of class `oligoSnpSet`, from scratch.

```
> ann[, "chromosome"] <- integer2chromosome(ann[, "chromosome"])
> fD <- new("AnnotatedDataFrame", data = ann, varMetadata = data.frame(labelDescription = colnames(ann
> pD <- annotatedDataFrameFrom(CT, byrow = FALSE)
> GT <- matrix(as.integer(GT), nrow(GT), ncol(GT))
> dimnames(GT) <- dimnames(CT)
> chromosome1 <- new("oligoSnpSet", copyNumber = CT, calls = GT,
+     featureData = fD, phenoData = pD, annotation = "pd.mapping50kHind.240,pd.mapping50kXba.240")
> validObject(chromosome1)

[1] TRUE

> options <- new("HmmOptions", snpset = chromosome1, states = states,
+     copyNumber.location = mu, copyNumber.scale = sample.sd[1],
+     probHomCall = c(0.5, 0.999, 0.7, 0.999, 0.7, 0.7))
> params <- new("HmmParameter", states = states(options),
+     initialStateProbability = 0.999)
> cn.emission <- copyNumber.emission(options)

[1] "Calculating emission probabilities on the log(copy number)"
[1] "User-supplied copyNumber.scale should be a standard deviation of the log2 CN"

> gt.emission <- calls.emission(options)
> emission(params) <- cn.emission + gt.emission
> genomicDistance(params) <- exp(-2 * diff(position(chromosome1))/(100 *
+     1e+06))
> transitionScale(params) <- matrix(1, length(states),
+     length(states))
> class(params)

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

> hmmpredict <- hmm(options, params)

[1] "Transforming copy number to log2 scale."
[1] "Fitting HMM to sample 1"

> class(hmmpredict)

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

> breaks <- findBreaks(predictions(hmmpredict), states = states,
+     position = ann[, "position"], chromosome = ann[,
+         "chromosome"], sample = colnames(CT))
```

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:

```
> gp <- plot(snpset(options), hmmpredict)

[1] "col.predict not specified in list of graphical parameters. Using the following colors:"
[1] "#8C510A" "#D8B365" "#F6E8C3" "#C7EAE5" "#5AB4AC" "#01665E"
```

```
> gp$abline.v <- TRUE
> allParameters <- unlist(snpPar(gp))
> gp$col.predict[3] <- "white"
> gp$hmm.ycoords <- c(0.7, 0.9)
> show(gp)
```



3.1 Integrating Confidence Estimates (ICE)

FIXME: integrating confidence estimates of the genotype calls, the probability of a missing genotype, epsilon probabiity for extreme observations

In this section, we illustrate how one may fit an HMM that incorporates confidence esimates of the SNP-level summaries for genotype calls and copy number. Confidence scores (inverse of standard errors) are available for this object (see Section 3.2 for how confidence scores were derived).

This information is incorporated into the HMM emission probabilities. Probably the easiest way to do that is recreate the options object, and then recalculate the emission probabilities.

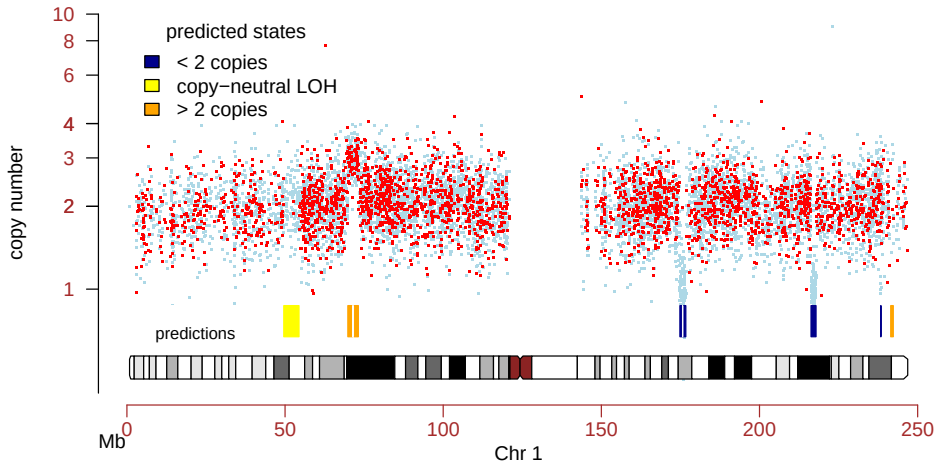
```
> data(chromosome1)
> range(cnConfidence(chromosome1))

[1] 0.2502008 6.9421269

> options <- new("HmmOptions", snpset = chromosome1, states = c("D",
+     "N", "L", "A"), copyNumber.location = c(1, 2, 2,
+     3), copyNumber.ICE = TRUE, probHomCall = c(0.99,
+     0.75, 0.99, 0.75))
> params <- new("HmmParameter", states = states(options),
+     initialStateProbability = 0.999)
> cn.emission <- copyNumber.emission(options)

[1] "Calculating emission probabilities on the log(copy number)"
[1] "Using 1/cnConfidence(object) as standard errors for the copy number"
```

```
> gt.emission <- calls.emission(options)
> emission(params) <- cn.emission + gt.emission
> genomicDistance(params) <- exp(-2 * diff(position(chromosome1))/(100 *
+     1e+06))
> transitionScale(params) <- matrix(1, length(states(options)),
+     length(states(options)))
> fit.ice <- hmm(options, params)

[1] "Transforming copy number to log2 scale."
[1] "Fitting HMM to sample 1"

> calculateBreakpoints(fit.ice)

    sample chr     start       end    nbases nprobes state
1  NA06993   1    836727  49545039  48708312     997     N
2  NA06993   1  49597810  54409755   4811945     102     L
3  NA06993   1  54498950  69838068  15339118     902     N
4  NA06993   1  69854466  71342708   1488242      97     A
5  NA06993   1  71406383  71474047     67664       4     N
6  NA06993   1  71826917  73174389   1347472     103     A
7  NA06993   1  73577300 174815096 101237796    3796     N
8  NA06993   1 174828535 175630040    801505      46     D
9  NA06993   1 175683520 175700199     16679       4     N
10 NA06993   1 175726310 176704067    977757      47     D
11 NA06993   1 176800399 216239917  39439518    1902     N
12 NA06993   1 216286002 217872810   1586808      99     D
13 NA06993   1 217892177 238302668  20410491     901     N
14 NA06993   1 238319943 238417864     97921       5     D
15 NA06993   1 238429864 241076215   2646351      92     N
16 NA06993   1 241483148 242295322    812174       7     A
17 NA06993   1 242374397 246860994   4486597      61     N
```

We may also incorporate the confidence scores for the genotype calls by specifying `calls.ICE=TRUE` (FIX ME). The slot `probHomCall` stores user-specified probabilities of P(call is AA or BB | state is LOH) and P(call is AA or BB | state is normal). These probabilities must be specified in this order. The emission probabilities for the genotype calls will only be calculated for the states LOH (LOH is defined as a stretch of homozygous genotype calls longer than what one would expect by chance) and Normal refers to typical ratios of heterozygous to homozygous genotype calls. The slot `term5` contains user-specified probabilities for the P(true genotype is HET | genotype call is AB, hidden state is Normal) and P(true genotype is HET | genotype call is AA or BB, hidden state is Normal), respectively. Default values are provided when not specified, as the following example illustrates.

```
> options <- new("HmmOptions", snpset = chromosome1, states = c("D",
+     "N", "L", "A"), copyNumber.location = c(1, 2, 2,
+     3), copyNumber.ICE = TRUE, calls.ICE = TRUE, probHomCall = c(0.99,
+     0.75))
> params <- new("HmmParameter", states = states(options),
+     initialStateProbability = 0.99)
> cn.emission <- copyNumber.emission(options)
> genomicDistance(params) <- exp(-2 * physicalDistance(options)/(100 *
+     1e+06))
> transitionScale(params) <- scaleTransitionProbability(states(options))
```

```
> gt.emit <- calls.emission(options)
> gt.emission <- array(NA, dim(cn.emission))
> gt.emission[, , 1:2] <- gt.emit
> gt.emission[, , 3:4] <- gt.emit
> emission(params) <- cn.emission + gt.emission
> fit.ice <- hmm(options, params)

> gp <- plot(snpset(options), fit.ice)

[1] "col.predict not specified in list of graphical parameters. Using the following colors:"
[1] "#A6611A" "white"   "#80CDC1" "#018571"

> gp$abline.v <- TRUE
> gp$col.predict <- c("darkblue", "white", "yellow", "orange")
> gp$hmm.coords

NULL

> show(gp)
> legend(-0.05, 10, fill = gp$col.predict[c(1, 3, 4)],
+     legend = c("< 2 copies", "copy-neutral LOH", "> 2 copies"),
+     bty = "n", title = "predicted states")
> legend(0, 0.8, legend = "predictions", bty = "n", cex = 0.8,
+     adj = 0)
```



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).

3.2 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 kernal based density estimates to obtain

$$f\left\{ \mathrm{S}_{\widehat{\mathrm{HOM}}} \mid \widehat{\mathrm{HOM}}, \mathrm{HOM} \right\},\ f\left\{ \mathrm{S}_{\widehat{\mathrm{HOM}}} \mid \widehat{\mathrm{HOM}}, \mathrm{HET} \right\},\ f\left\{ \mathrm{S}_{\widehat{\mathrm{HET}}} \mid \widehat{\mathrm{HET}}, \mathrm{HOM} \right\},\quad \text{and } f\left\{ \mathrm{S}_{\widehat{\mathrm{HET}}} \mid \widehat{\mathrm{HET}}, \mathrm{HET} \right\} \tag{1}$$

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 ($\widehat{\mathrm{HOM}}$) 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: $\mathrm{Gamma}(1, 2) + 0.3$, 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 $\widehat{\mathrm{CN}}$ 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*.

4 The HmmParameter class

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

```
> new("HmmParameter")
```

The object `params` contains all of the parameters needed for fitting the HMM, including an estimate of the genomic distance between SNPs (used for calculating SNP-specific transition probabilities), emission probabilities (slot: emission), and initial state probabilities.

Emission probabilities. The emission probabilities are stored as an array in the `params` object. The emission probability array has dimension R $\times$ C $\times$ S, where S is the number of hidden states, R is the number of rows (SNPs), and C is the number of samples. One may use [to subset object of class `HmmParameter`.

```
> params[5, 1, ]
```

```
Formal class 'HmmParameter' [package "VanillaICE"] with 5 slots
  ..@ states                 : chr [1:4] "D" "N" "L" "A"
  ..@ initialStateProbability: num [1:4] 0.000333 0.999 0.000333 0.000333
  ..@ emission               : num [1, 1, 1:4] -1.2 -1.47 -1.19 -1.79
  .. ..- attr(*, "dimnames")=List of 3
  .. .. ..$ : chr "SNP_A-1662392"
  .. .. ..$ : chr "NA06993"
  .. .. ..$ : chr [1:4] "D" "N" "L" "A"
  ..@ genomicDistance        : num 1
  ..@ transitionScale        : num [1:4, 1:4] 1 1 1 1 1 1 1 1 1 1 ...
```

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: $e^{-2d(100*1e6)}$. 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. The SNP-specific transition probabilities can be scaled by specifying a matrix. No scaling of the transition probabilities between states, the default, occurs when the scaling matrix is all 1's:

```
> transitionScale(params)

     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    1    1    1    1
[3,]    1    1    1    1
[4,]    1    1    1    1
```

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 commmand:

```
> transitionScale(params) <- scaleTransitionProbability(states(params),
+     SCALE = 10)
> transitionScale(params)
```

5 The HmmOptions Class

To be completed ...

6 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.ice

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:
    dbsnp_rs_id: dbsnp_rs_id
    chromosome: chrom
    ...: ...
    arm: NA
    (9 total)
experimentData: use 'experimentData(object)'
```

```
Annotation: pd.mapping50k.hind240,pd.mapping50k.xba240
hidden states:  D N L A
breakpoints:
'data.frame':        17 obs. of  7 variables:
 $ sample : chr  "NA06993" "NA06993" "NA06993" "NA06993" ...
 $ chr    : chr  "1" "1" "1" "1" ...
 $ start  : int  836727 49597810 54498950 69854466 71406383 71826917 73577300 174828535 175683520 1757
 $ end    : int  49545039 54409755 69838068 71342708 71474047 73174389 174815096 175630040 175700199 1
 $ nbases : int  48708312 4811945 15339118 1488242 67664 1347472 101237796 801505 16679 977757 ...
 $ nprobes: int  997 102 902 97 4 103 3796 46 4 47 ...
 $ state  : chr  "N" "L" "N" "A" ...
```

The breakpoints are stored in slot `breakpoints` of an object of class `HmmPredict`. These functions do roughly the same thing:

```
> breaks <- breakpoints(fit.ice)
> predict <- predictions(fit.ice)
> breaks <- findBreaks(x = predict, states = states(fit.ice),
+     position = position(fit.ice), chromosome = chromosome(fit.ice),
+     sample = sampleNames(fit.ice))
> breaks <- calculateBreakpoints(fit.ice)
```

7 HMMs for different classes of data

7.1 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")
> options <- new("HmmOptions", snpset = chr1.cn, states = c("D",
+     "N", "A"), copyNumber.location = 1:3)
> params.cn <- new("HmmParameter", states = c("D", "N",
+     "A"))
> emission(params.cn) <- copyNumber.emission(options)

[1] "Calculating emission probabilities on the log(copy number)"

> transitionScale(params.cn) <- scaleTransitionProbability(states = c("D",
+     "N", "A"), normalLabel = "N")
> genomicDistance(params.cn) <- exp(-2 * physicalDistance(options)/(100 *
+     1e+06))
> fit.cn <- hmm(options, params.cn)

[1] "Transforming copy number to log2 scale."
[1] "Fitting HMM to sample 1"

> breakpoints(fit.cn)
```

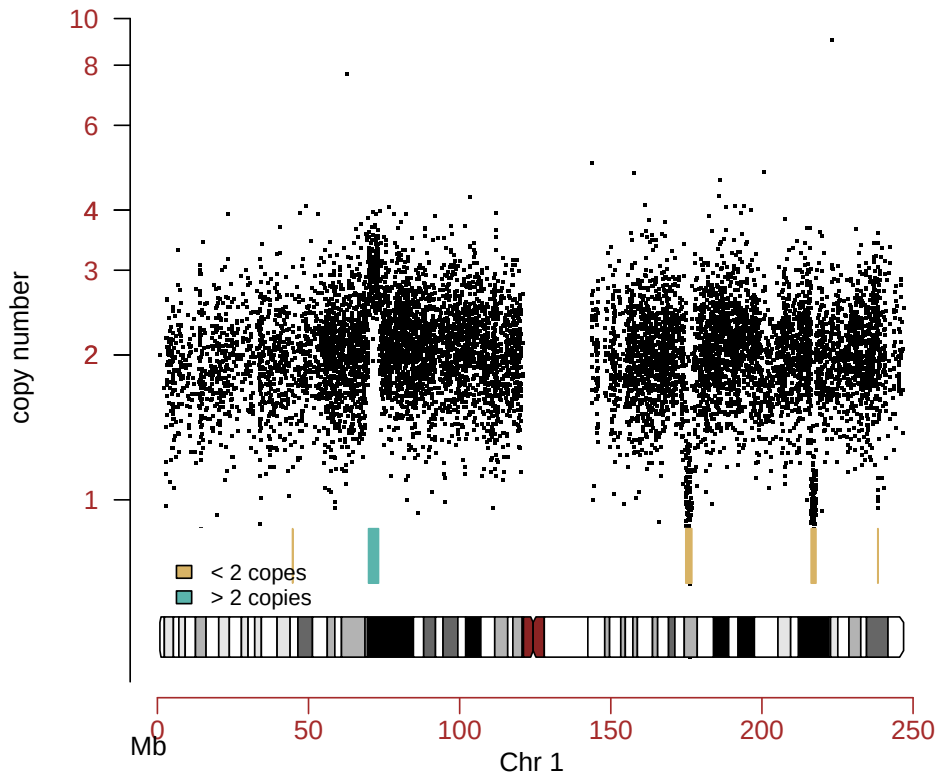
```
    sample chr     start       end    nbases nprobes state
1  NA06993   1    836727  44702452  43865725     877     N
2  NA06993   1  44722116  44762242     40126       2     D
3  NA06993   1  44779351  69838068  25058717    1122     N
4  NA06993   1  69854466  73153900   3299434     202     A
5  NA06993   1  73174070 174814292 101640222    3797     N
6  NA06993   1 174815096 176800844   1985748     101     D
7  NA06993   1 176926556 216239917  39313361    1899     N
8  NA06993   1 216286002 217872810   1586808      99     D
9  NA06993   1 217892177 238302668  20410491     901     N
10 NA06993   1 238319943 238417864     97921       5     D
11 NA06993   1 238429864 246860994   8431130     160     N
```

```
> graph.par <- plot(snpset(options), fit.cn)
```

```
[1] "col.predict not specified in list of graphical parameters. Using the following colors:"
[1] "#D8B365" "white"   "#5AB4AC"
```

```
> graph.par$abline.v <- FALSE
```

```
> print(graph.par)
> legend(0, 0.8, fill = graph.par$col.predict[c(1, 3)],
+     legend = c("< 2 copes", "> 2 copies"), bty = "n",
+     cex = 0.8)
```

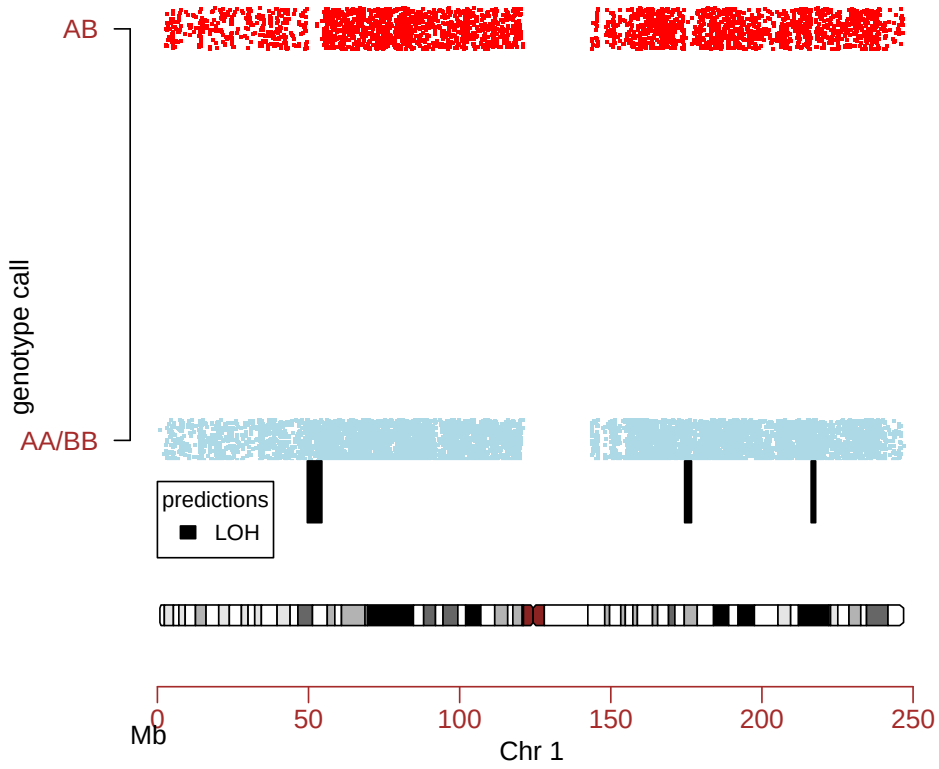


7.2 Genotype calls

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 indvidual 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")
> options.calls <- new("HmmOptions", snpset = chr1.calls,
+     states = c("L", "N"), probHomCall = c(0.99, 0.7))
> params.calls <- new("HmmParameter", states = states(options.calls))
> transitionScale(params.calls) <- scaleTransitionProbability(states(options.calls))
> genomicDistance(params.calls) <- exp(-2 * physicalDistance(options.calls)/(100 *
+     1e+06))
> emission(params.calls) <- calls.emission(options.calls)
> fit.calls <- hmm(options.calls, params.calls)

[1] "Fitting HMM to sample 1"
```

```
> breakpoints(fit.calls)

   sample chr     start       end    nbases nprobes state
1 NA06993   1    836727  49545039  48708312     997     N
2 NA06993   1  49597810  54409755   4811945     102     L
3 NA06993   1  54498950 174309008 119810058    4890     N
4 NA06993   1 174418117 176704067   2285950     109     L
5 NA06993   1 176800399 216239917  39439518    1902     N
6 NA06993   1 216286002 217771828   1485826      97     L
7 NA06993   1 217872013 246860994  28988981    1068     N

> gp <- plot(snpset(options.calls), fit.calls)

[1] "col.predict not specified in list of graphical parameters. Using the following colors:"
[1] "black" "white"

> gp$col.predict <- c("black", "white")
> gp$ylim <- c(-0.5, 1)
> gp$add.centromere <- FALSE
> gp$abline.v <- TRUE
> gp$cytoband.ycoords <- c(-0.45, -0.4)
> gp$hmm.ycoords <- c(-0.2, -0.05)

> print(gp)
> legend(0, -0.1, legend = "LOH", fill = "black", title = "predictions",
+     bty = "o", cex = 0.8)
```



7.3 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*.

8 Session Information

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

- R version 2.8.0 Under development (unstable) (2008-06-18 r45949), `powerpc-apple-darwin8.11.0`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, splines, stats, tools, utils
- Other packages: Biobase 2.1.0, RColorBrewer 1.0-1, SNPchip 1.5.2, VanillaICE 1.3.8, genefilter 1.15.10, oligoClasses 1.3.8, survival 2.32
- Loaded via a namespace (and not attached): AnnotationDbi 1.3.8, DBI 0.2-4, RSQLite 0.6-4, annotate 1.15.6

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.