Notes on CNV Methods

Samuel K. Lee* and David Benjamin†
Broad Institute, 75 Ames Street, Cambridge, MA 02142

(Dated: January 11, 2016)

Some notes on current and proposed methods used in the GATK CNV workflow.

I. INTRODUCTION

The GATK uses two types of information from sequencing data to detect copy number variations (CNVs). First, targets (usually exons but in principle any genomic locus) with abnormally high or low coverage suggest amplifications or deletions, respectively. Second, sites that are heterozygous in a normal sample and have allele ration significantly different from 1:1 in the matched tumor sample imply a CNV event involving one or both alleles. The workflow is as follows:

1. Partition targets into continuous segments that represent the same copy-number event using coverage data. The segmentation is performed by a circular-binary-segmentation (CBS) algorithm described by Olshen et al. 2004 that was originally developed to segment noisy array copy-number data.[1]

2. Find heterozygous sites in the normal case sample and segment these, again using CBS, according to their ref:alt allele ratios in the tumor sample.

3. Combine the two sets of segments in a liberal manner that tends to produce too many segments.

4. Alternate between modeling the copy ratio and minor allele fraction of each segment with merging adjacent segments that are sufficiently similar according to this model.

II. SEGMENTATION BY COVERAGE AND MINOR ALLELE FRACTION

A. Panel of Normals

We cannot simply divide the coverage of each target by the average sequencing depth to obtain an estimate of its copy ratio. The coverage of different targets is heavily-biased by factors including the efficiency of their baits, GC content, and mappability. In order to detect CNVs we must model the coverage of each target in the absence of CNVs, which requires a panel of normal samples (PoN) that are representative of the sequencing conditions of the case sample. PoN samples must be created using the same baits as the case sample. The steps for creating a panel of normals are

1. Obtain the coverage (total number of overlapping reads) of every target and sample.

2. Calculate the median coverage of each target over all samples.

3. Filter out targets whose median coverage is below a given percentile (by default 25%) of target medians.

4. Divide all coverages by their corresponding target medians.

5. Filter out samples with too great a proportion of zero-coverage targets (by default 5%).

6. Filter out targets with zero coverage in too great a proportion of samples (by default 2%).

7. Filter out samples whose median coverage is above or below certain percentiles (by default 2.5% and 97.5%) of sample medians.

*Electronic address: `slee@broadinstitute.org`
†Electronic address: `davidben@broadinstitute.org`

[1] Specifically, the CBS implementation provided by the `R` package `DNACopy` is used.

8. Replace all remaining zero coverages with their corresponding target median.

9. Calculate the range of coverage from percentile $p\%$ to $(100-p)\%$ for each target and truncate coverages at each target to lie within these ranges. By default $p = 0.1$.

10. Divide each coverage by its sample median.

11. Take the $\log_2$ of each coverage.

12. Calculate the median of each sample and take the median of these over all targets. Subtract this median of medians from each coverage.

13. Perform a singular value decomposition (SVD) of the resulting matrix and calculate its pseudo-inverse truncated to the space spanned by the k right eigenvectors with largest singular values. Choose k using Jollife's heuristic of retaining singular values greater than 0.7 times the mean singular value.

The output is: a $N \times k$ matrix P, the columns of which are the the retained right eigenvectors (eigensamples), and its pseudoinverse P^+; and the target medians (before any transformations). Here N denotes the number of targets.

B. Normalizing case samples

We first divide the integer coverage of the case sample at each target by the corresponding target median from the PoN and take the $\log_2$ transformation to obtain an $N \times 1$ column matrix $\mathbf{x}$. We then calculate the "tangent-normalized" coverage: $\mathbf{x} - PP^+\mathbf{x}$. The meaning of this is as follows: PP^+ is an operator that projects onto the column space of P. That is, it projects onto the space spanned by the k most significant eigensamples representing the (non-CNV-related) variability of the coverage. Subtracting the projection $PP^+\mathbf{x}$ therefore isolates the CNV signal and removes noise due to fluctuations in sequencing bias.

Finally, the tangent-normalized coverage vector is passed to CBS to obtain coverage segments.

C. Segmentation by Minor Allele Fraction

Given a large database of common SNPs, we search the normal case sample for heterozygous sites. To determine whether a site with r ref reads and a alt reads is heterozygous, we calculate the two-sided p-value under the null hypothesis that the number of alt reads follows a binomial distribution: $a \sim \mathrm{Binom}(a + r, 1/2)$. If the p-value is not too small we consider the site heterozygous.

To obtain allele fraction segments, we estimate the minor allele fraction of a het site as $\min(a, r)/(a + r)$, where a and r are counts for the tumor case sample, and pass the resulting list to CBS.

D. Target/SNP segment union

[SL: DB can fill this in.]

E. Small-segment merging

[SL: SL to update this to the new method.]

Using CBS to segment the targets in `ReCapSeg` results in segments that are [SL: always?] larger than ∼2–3 targets. However, after taking the union of target and SNP segments, small segments with less than ∼2–3 targets may be introduced. To be consistent with CBS and `ReCapSeg`, `AllelicCapSeg` treats these small segments as spurious, and removes them by merging them with adjacent segments.

A segment is considered to be small if the number of targets it contains is strictly less than a threshold number of targets n_t; we take $n_t = 3$. The small-segment merging algorithm checks each ith segment in turn, starting with the first, leftmost segment. If the segment is small, it is repeatedly merged with the adjacent segment that is closer in the L_1 distance $|\tau_i - \tau_{i\pm1}| + |f_i - f_{i\pm1}|$ in copy-ratio–allele-fraction space until it is no longer small.[2] Exceptions occur for adjacent segments on different chromosomes, which are never merged; in practice, this is enforced by setting the L_1 distance between segments on different chromosomes to be infinite. After all segments have been checked and merged, any remaining small segments (which will be present if any chromosome contains less than n_t targets) are dropped.

III. CURRENT HELLBENDER `AllelicCapSeg` WORKFLOW

A. Segmented model

We want a generative model for allelic fractions that infers its parameters from the data. We observe alt and ref read counts for each het site and wish to infer the minor allelic fraction of every segment. Let's consider what other hidden variables belong in the model. Read counts obey an overdispersed binomial distribution in which the probability of an alt read is a site-dependent random variable. Letting θ_j be the probability that a mapped read at het j is an alt we have

$$P(a_j, r_j|\theta_j) = \binom{a_j + r_j}{a_j} \theta_j^{a_j} (1-\theta_j)^{r_j} = \binom{n_j}{a_j} \theta_j^{a_j} (1-\theta_j)^{r_j}, \tag{1}$$

where a_j and r_j are alt and ref read counts and $n_j = a_j + r_j$ is the total read count at site j. Now we consider θ_j. Suppose site j belongs to a segment with minor allelic fraction f and is alt minor, such that $P(\text{alt}) = f$ and $P(\text{ref}) = 1 - f$ are the probabilities that a random DNA fragment will contain the alt and ref alleles. Let $x_j^{\text{alt(ref)}} = P(\text{mapped}|\text{alt(ref)})$ be the probabilities that an alt (ref) DNA fragment at site j eventually gets sequenced and mapped. Then θ_j is the conditional probability that a mapped read comes from an alt fragment:

$$\theta_j = P(\text{alt}|\text{mapped}) = \frac{P(\text{alt})P(\text{mapped}|\text{alt})}{P(\text{alt})P(\text{mapped}|\text{alt}) + P(\text{ref})P(\text{mapped}|\text{ref})} \tag{2}$$

$$= \frac{f x_j^{\text{alt}}}{f x^{\text{alt}} + (1-f) x_j^{\text{ref}}} = \frac{f}{f + (1-f)\lambda_j}, \tag{3}$$

where $\lambda_j = x_j^{\text{ref}} / x_j^{\text{alt}}$ is the "bias ratio" of ref to alt sequenceability and mappability at site j. A similar result for ref minor sites follows from substituting $f \leftrightarrow 1 - f$. In addition to the bias ratio λ_j we need an indicator variables z_j with three states, alt minor, ref minor, and an outlier state that gives robustness to anomalous events. For this outlier state we average the binomial likelihood over all θ to get:

$$P(a_j, r_j|\text{outlier}) = \binom{n_j}{a_j} \int_0^1 \theta_j^{a_j} (1-\theta_j)^{r_j} \, d\theta_j = \binom{n_j}{a_j} \frac{a_j! r_j!}{(n_j+1)!} \tag{4}$$

For notational convenience we give z_j a one-of-K encoding $z_j = (z_{ja}, z_{jr}, z_{jo})$ in which one component equals 1 and the rest 0.

The contribution of site j to the likelihood is

$$P(a_j, r_j|f_j, \lambda_j, z_j) = \binom{n_j}{a_j} \left[\frac{f_j^{a_j} (1-f_j)^{r_j} \lambda_j^{r_j}}{(f_j + (1-f_j)\lambda_j)^{n_j}} \right]^{z_{ja}} \left[\frac{(1-f_j)^{a_j} f_j^{r_j} \lambda_j^{r_j}}{(1 - f_j + f_j \lambda_j)^{n_j}} \right]^{z_{jr}} \left[\frac{a_j! r_j!}{(n_j+1)!} \right]^{z_{jo}} \tag{5}$$

where f_s is the minor allele fraction of the segment containing site j. We will consider f to be drawn from a uniform distribution on $[0, 1/2]$ – that is, we give it a flat prior – but in the future we can obtain some sort of clustering behavior, representing the fact that events in the same subclone that exhibit the same integer copy numbers will have the same minor allelic fractions, by drawing f_s from a Dirichlet process.

We assume that the bias ratios come from a common Gamma distribution with parameters α, β:

$$P(\lambda_j|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_j^{\alpha-1} e^{-\beta\lambda_j} \tag{6}$$

[2] To be explicit, segments are reindexed after each merge, so that the new segment formed by merging segment i and segment $i \pm 1$ retains the index i.

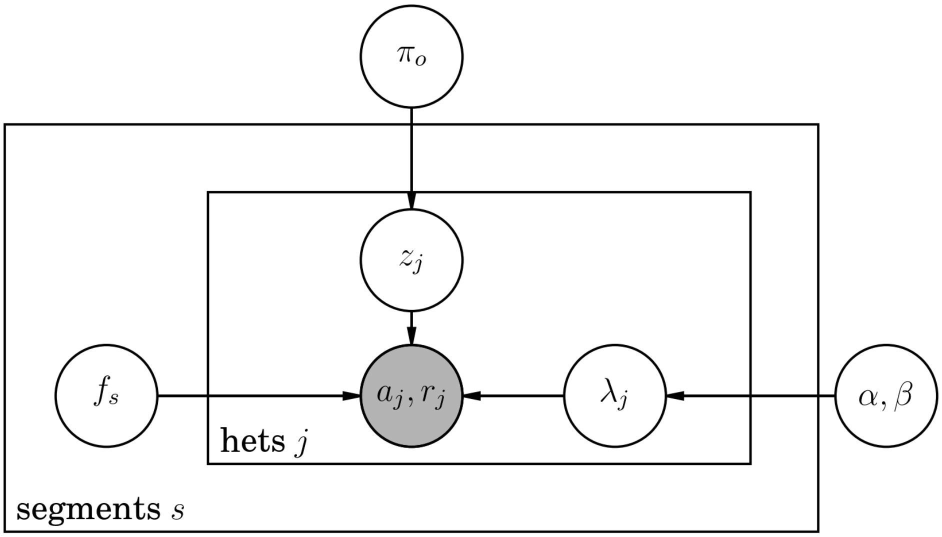

FIG. 1: Graphical model for `AllelicCapSeg`

Note that bias ratios tend to be near 1.0 and so the choice of distribution is not too important as long as it has adjustable mean and standard deviation. We choose the Gamma distribution because it is the simplest such distribution on $\mathbb{R}^+$. We will give the parameters α and β a flat prior $P(\alpha, \beta) \propto 1$.

Finally, the indicator z_j is a multinomial random variable distributed according to parameter vector π:

$$P(z_{ja(r,o)} = 1|\pi) = \pi_{a(r,o)} \tag{7}$$

We set the alt and ref minor probabilities equal so that the only free parameter is $\pi = \pi_o$, with $\pi_{a(r)} = (1 - \pi)/2$. The Bayesian network corresponding to this model is shown in Figure III A.

As with the other parameters, we put a flat prior on π. Putting all the pieces together the likelihood is

$$\mathbb{L} = \prod_j \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_j^{\alpha-1} e^{-\beta\lambda_j} \left[\frac{(1-\pi) f_j^{a_j} (1-f_j)^{r_j} \lambda_j^{r_j}}{(f_j + (1-f_j)\lambda_j)^{n_j}} \right]^{z_{ja}} \left[\frac{(1-\pi)(1-f_j)^{a_j} f_j^{r_j} \lambda_j^{r_j}}{(1-f_j+f_j\lambda_j)^{n_j}} \right]^{z_{jr}} \left[\frac{2\pi a_j! r_j!}{(n_j+1)!} \right]^{z_{jo}}. \tag{8}$$

The dependence on λ for alt minor sites is

$$g(\lambda_j, \alpha, \beta, f_j, a_j, r_j) = \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{f_j^{a_j} (1-f_j)^{r_j} \lambda_j^{\alpha+r_j-1} e^{-\beta\lambda_j}}{(f_j + (1-f_j)\lambda_j)^{n_j}}. \tag{9}$$

For ref minor sites the dependence is the same but with $f \leftrightarrow 1-f$. We show in show in Appendix A that this function can be integrated analytically, and thus we can marginalize λ out of the model to obtain the likelihood

$$\prod_j \left[\frac{1-\pi}{2} \phi(\alpha, \beta, f_j, a_j, r_j) \right]^{z_{ja}} \left[\frac{1-\pi}{2} \phi(\alpha, \beta, 1-f_j, a_j, r_j) \right]^{z_{jr}} \left[\frac{\pi a_j! r_j!}{(n_j+1)!} \right]^{z_{jo}}, \tag{10}$$

where $\phi(\alpha, \beta, f_j, a_j, r_j) = \int_0^\infty g(\lambda, \alpha, \beta, f, a, r)\, d\lambda_j$. Pseudocode for computing ϕ is presented in Appendix A. Furthermore, marginalizing out z is trivial – simply sum each term over its three possible states. We then have a collapsed likelihood

$$p(f, \alpha, \beta, \pi) \propto \prod_j \left[\frac{1-\pi}{2} \phi(\alpha, \beta, f_j, a_j, r_j) + \frac{1-\pi}{2} \phi(\alpha, \beta, 1-f_j, a_j, r_j) + \frac{\pi a_j! r_j!}{(n_j+1)!} \right] \tag{11}$$

Integrating out the latent variables removes the strongest correlations from the model – intuitively, f should not be too sensitive to α and β, for example – and significantly improves mixing. The exception is α and β, since adjusting one with the other fixed changes the mean of the prior on λ. Thus we reparameterize in terms of μ and σ^2, the mean and variance of the common gamma distribution of biases, where $\alpha = \mu^2/\sigma^2$ and $\beta = \mu/\sigma^2$. Due to the weak correlations our MCMC method does not need to be very sophisticated. We choose to sample each variable with one-dimensional adaptive Metropolis, tuning the proposal step size to achieve some reasonable acceptance rate like 0.4 or so. Thus we have completely specified an MCMC scheme for this model, given by Algorithm 1:

Algorithm 1 MCMC algorithm for `AllelicCapSeg`

```
Initialize all parameters to a maximum likelihood initial guess (see below).
repeat
    Sample each $f_s$ with adaptive Metropolis
    Sample $\pi$ with adaptive Metropolis
    Sample $\mu$ with adaptive Metropolis
    Sample $\beta$ with adaptive Metropolis
until convergence
```

We initialize the model by finding the mode of likelihood. This significantly reduces burn-in time of our MCMC sampling. Also, it allows us to give the adaptive Metropolis samplers better initial guesses for their step sizes. Since in practice there is a single global maximum of the likelihood it is easy to find. After initializing the initialization with rough guesses for the parameters, we successively find one-dimensional maxima adjusting one parameter at a time until the likelihood converges. One could use multidimensional optimization to obtain faster convergence, but after marginalizing out latent parameters the remaining correlations are weak and thus this simple approach performs quite well. Since we may delegate one-dimensional maximization to mathematical libraries, the only thing left to describe is our initial coarse guess.

In the initial guess we set the outlier probability $\pi_o = 0.01$, $\mu = 1.0$, and $\sigma^2 = 0.1$. With the exception of σ^2 these are all reasonable guesses. We choose σ^2 to be larger than what we actually believe because μ converges more slowly from a bad initial guess if σ^2 is too small. The only non-trivial part of the initial guess is the minor allele fractions. For each segment, we wish to set the minor allele fraction to the number of reads from minor alleles divided by to total number of reads – this is an unbiased estimator if allelic bias is absent. The problem is that we have counts of alt and ref reads, not minor and major reads. Our solution is to weight the alt and ref read counts on each het by probabilities that the het is alt and ref minor, respectively. That is, we set

$$f_S \approx \frac{\sum_{j \in S} a_j P(z_{ja} = 1) + r_j P(z_{jr} = 1)}{\sum_{j \in S} (a_j + r_j)(P(z_{ja} = 1) + P(z_{jr} = 1))} \tag{12}$$

For this coarse guess we ignore the possibility of outliers, so that $P(z_{ja} = 1) + P(z_{jr} = 1) = 1$. Ignoring bias and outliers the alt minor likelihood of het j is proportional to $f_j^{a_j}(1 - f_j)^{r_j}$. Since we don't know f yet, we integrate this (including the normalization) from $f = 0$ to $f = 1/2$ in order to get $P(z_{ja} = 1)$. This quantity is called the incomplete regularized beta function I. Thus we have

$$P(z_{ja} = 1) \approx I(1/2, a_j + 1, r_j + 1), \quad P(z_{jr} = 1) = 1 - P(z_{ja} = 1). \tag{13}$$

B. Target/SNP segment union

C. Similar-segment merging

IV. PROPOSED METHODS FOR `hellbender`

A. Using Panel of Normals for Allelic Fraction Model

The `AllelicCapSeg` model learns a global distribution on allelic biases and uses it as a shared prior for the allelic biases of SNPs. While better than nothing, it would be much more powerful to use prior knowledge of the allelic bias at each SNP individually. We can learn these per-SNP biases from a panel of normals using the `AllelicCapSeg` model, but with two simplifications. First, minor allele fractions are always 1/2 since normal samples are diploid and do not exhibit subclonality. Second, we do not account for outliers; that is, we set the outlier probability $\pi = 0$. The reason for this is that the panel of normals reflects typical distributions of allelic biases and censoring data via an outlier classification could render these distributions artificially tight. If the allelic bias at some SNP site varies a lot we want to know about it. The overall likelihood is

$$\prod_j \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_j^{\alpha-1} e^{-\beta\lambda_j} \prod_{s\in\mathcal{H}_j} \frac{\lambda_j^{r_{sj}}}{(1+\lambda_j)^{n_{sj}}} \tag{14}$$

$$= \prod_j \frac{\beta^\alpha}{\Gamma(\alpha)} e^{-\beta\lambda_j} \frac{\lambda_j^{\alpha+r_{\cdot j}-1}}{(1+\lambda_j)^{n_{\cdot j}}} \tag{15}$$

where λ_j is the allelic bias ratio of SNP j (for samples sequenced and mapped using the same technology as the panel of normals), $\mathcal{H}_j$ is the set of samples in the panel of normals that are heterozygous at SNP j, $r_{\cdot j} = \sum_{s\in\mathcal{H}_j} r_{sj}$, and $n_{\cdot j} = \sum_{s\in\mathcal{H}_j} n_{sj}$. As before, the biases are assumed to come from a common distribution Gamma(α,β), but due to the large number of samples in the panel of normals the data will yield a posterior distribution on each λ_j that may be quite different from the global prior. It is these posteriors that we will use as input to `AllelicCapSeg`. Although they are the object of interest, however, we will first marginalize them out of the likelihood in order to obtain maximum likelihood estimates of α and β. We have in fact already performed this marginalization – Equation 15 is the special case $f = 1/2$, $\pi = 0$ of the `AllelicCapSeg` likelihood, Equation 8, and thus its marginalization over latent variables is obtained by substituting $f = 1/2$, $\pi = 0$ into Equation 11, which yields

$$p(\alpha,\beta) = \prod_j \phi(\alpha,\beta,f=1/2,n_{\cdot j}-r_{\cdot j},r_{\cdot j}). \tag{16}$$

This likelihood is easily maximized numerically to obtain MLE values of α and β. Having done this, we can then approximate the posterior on each λ_j as a gamma distribution using the method of Appendix A. As shown there, the posterior on λ_j is Gamma(ρ_j,τ_j) where ρ_j and τ_j are computed in Algorithm 3, with $a \to n_{\cdot j} - r_{\cdot j}$ and $r \to r_{\cdot j}$.

Once we have the posteriors on each λ_j from the panel of normals, they are used as priors for λ_j in the `AllelicCapSeg` model. This obviates the hyperparameters α and β, and Equation 11 becomes

$$p(f,\pi) \propto \prod_j \left[\frac{1-\pi}{2}\phi(\rho_j,\tau_j,f_j,a_j,r_j) + \frac{1-\pi}{2}\phi(\rho_j,\tau_j,1-f_j,a_j,r_j) + \frac{\pi a_j! r_j!}{(n_j+1)!} \right] \tag{17}$$

where f and π may once again be sampled via adaptive Metropolis.

B. Bayesian het pulldown

We are given a large data set of ref and alt read counts over many potential SNP sites and we wish to infer which sites are hets and with what probabilities. This problem is naturally represented as a mixture model in which the hidden labels are genotypes – hom ref, het, and hom alt. Since the observed data are ref and alt counts is seems natural to use a binomial mixture model in which the binomial parameters are the probability of an alt read. Then the binomial parameters are the error rate for hom ref genotypes, 1/2 times the allelic bias for het genotypes, and 1 minus the error rate for hom alt genotypes. However, actual data are overdispersed because the error rate and allelic bias are random variables, not single parameters. For example, sequencing error rates and allelic bias (for concreteness, consider mapping bias) depend on context. Thus a beta-binomial mixture model is more appropriate. A maximum-likelihood (MLE) approach will yield posterior probabilities on the genotypes at each locus, in particular the het probability. It also gives the parameters of a beta distribution of allelic bias, which is useful downstream in ACS.

For generality and at no cost in complexity, consider a Dirichlet-multinomial mixture (DMM) with K mixture components and M classes of observed data. For our purposes there are $K = 3$ genotypes and $M = 2$ types of read, ref and alt. The beta-binomial distribution is the $M = 2$ case of the Dirichlet-multinomial. The observed data are counts n_{jm}, the number of times class m was seen in observation j. For us, each potential SNP site is a datum j. Let $N_j = \sum_m n_{jm}$ denote the total read count at site j. For our purposes, $\{N_j\}$ are constants – we are not trying to model depth of coverage here, just the proportions of the coverage allotted to ref and alt reads.

We represent hidden labels via the 1-of-K encoding $\mathbf{z}_j = (0,0,\ldots 1,0,0\ldots)$, so $z_{jk} = 1$ when datum j comes from component k. The hidden labels are multinomially distributed as $P(\mathbf{z}_j) = \text{Mult}(\mathbf{z}_j|\pi)$, where π_k is the probability of component k and $\sum_k \pi_k = 1$. Finally, the observed counts for mixture component k are drawn from a Dirichlet-multinomial distribution with parameter α_k:

$$P(\mathbf{n}_j \mid z_{jk}=1,\alpha_k) = \frac{\Gamma(A_k)}{\Gamma(A_k+N_j)}\prod_m \frac{\Gamma(\alpha_{km}+n_{jm})}{\Gamma(\alpha_{km})}, \tag{18}$$

where $A_k = \sum_m \alpha_{km}$.

The EM algorithm for MLE estimates of $\{\pi_k\}$ and $\{\alpha_{km}\}$ requires the complete-data likelihood (CDL), that is, the joint likelihood of the observed data and hidden labels given the parameters. In contrast, a direct approach maximizes the likelihood marginalized over the hidden variables. The CDL of of the DMM is

$$P(\mathbf{z},\mathbf{n} \mid \pi,\alpha) = P(\mathbf{z} \mid \pi)P(\mathbf{n} \mid \mathbf{z},\alpha) \tag{19}$$

$$= \prod_{jk}\left[\pi_k \frac{\Gamma(A_k)}{\Gamma(A_k+N_j)}\prod_m \frac{\Gamma(\alpha_{km}+n_{jm})}{\Gamma(\alpha_{km})}\right]^{z_{jk}} \tag{20}$$

In the E step of the EM algorithm, we obtain the posterior distribution on $P(\mathbf{z} \mid \mathbf{n},\pi,\alpha)$ from Eq. (20). By inspection the posterior is a product of independent multinomials

$$\bar{z}_{jk} \equiv P(z_{jk}=1 \mid \mathbf{n},\pi,\alpha) \propto \pi_k \frac{\Gamma(A_k)}{\Gamma(A_k+N_j)}\prod_m \frac{\Gamma(\alpha_{km}+n_{jm})}{\Gamma(\alpha_{km})}, \tag{21}$$

with a normalization constant determined by the condition $\sum_k \bar{z}_{jk} = 1$.

In the M step of the EM algorithm we take the expectation of the log-CDL with respect to the posterior on $\mathbf{z}$ and maximize with respect to π and α. That is, we maximize

$$\sum_{jk} \bar{z}_{jk}\left\{\log \pi_k + \log \frac{\Gamma(A_k)}{\Gamma(A_k+N_j)} + \sum_m \log \frac{\Gamma(\alpha_{km}+n_{jm})}{\Gamma(\alpha_{km})}\right\}. \tag{22}$$

Maximizing with respect to π_k with a Lagrange multiplier for the constraint $\sum_k \pi_k = 1$

$$\pi_k = \frac{\sum_j \bar{z}_{jk}}{\sum_{j\ell} \bar{z}_{j\ell}} \tag{23}$$

To maximize with respect to α we use the fact that if we are trying to maximize $f(\mathbf{x})$ and have a current guess of $\mathbf{x}_0$, then an improved guess may be obtained by maximizing $g(\mathbf{x})$, where $g(\mathbf{x}_0) = f(\mathbf{x}_0)$ and $g(\mathbf{x}) \leq f(\mathbf{x})$ for all $\mathbf{x}$. Furthermore, repeating this gives an iterative optimization that converges to a local maximum. Using bounds (B7) and (B9) and dropping additive constants, we find that the iterative step is to maximize the lower bound

$$\sum_{jk} \bar{z}_{jk}\left\{-\left(\psi(\hat{A}_k+N_j)-\psi(\hat{A}_k)\right)A_k + \sum_m \hat{\alpha}_{km}\left(\psi(\hat{\alpha}_{km}+n_{jm})-\psi(\hat{\alpha}_{km})\right)\log(\alpha_{km})\right\}. \tag{24}$$

with respect to α_{km} treating the "old" guesses $\hat{\alpha}_{km}$ as constants. This maximization is a straightforward matter of setting the derivative to zero and gives the fixed-point iteration

$$\alpha_{km} = \hat{\alpha}_{km}\frac{\sum_j \bar{z}_{jk}\left(\psi(\hat{\alpha}_{km}+n_{jm})-\psi(\hat{\alpha}_{km})\right)}{\sum_j \bar{z}_{jk}\left(\psi(\hat{A}_k+N_j)-\psi(\hat{A}_k)\right)} \tag{25}$$

As is often the case with mixture models, we risk converging to a bad local maximum if parameters are initialized poorly. Following the approach used by Thomas Minka in his FastFit software, we obtain a good initial guess by fitting a Dirichlet mixture model (as opposed to a Dirichlet-multinomial model) on effective multinomial pseudodata. That is, instead of working with *counts* n_{jm}, work with *proportions* $p_{jm} = n_{jm}/N_j$. Since $\sum_m p_{jm} = 1$, $\mathbf{p}_j$ can be interpreted as a multinomial distribution drawn from a Dirichlet mixture. This preprocessing step maps the original count data onto the $(M-1)$-dimensional simplex, and we can then assign the pseudo-multinomials $\{\mathbf{p}_j\}$ to K clusters via the K-means algorithm. Define the indicator variable $\chi_{jk} = 1$ if pseudo-multinomial $\mathbf{p}_j$ is assigned to cluster k and let $\chi_{jk} = 0$ otherwise.

We initialize π_k as the empirical proportion of mixture component k in the clustering step. That is

$$\pi_k = \frac{\sum_j \chi_{jk}}{N} = \frac{N_k}{N} \tag{26}$$

where N_k is the number of pseudo-multinomials assigned to cluster k.

Then for each component k we initialize the Dirichlet parameter vector $\alpha_\mathbf{k}$ via moment matching. Parameterize $\alpha_\mathbf{k}$ as $\alpha_k = s_k \theta_k$, where $\sum_m \theta_{km} = 1$ is the mean of the Dirichlet distribution and s is its concentration. Since multinomials $S_k = \{\mathbf{p}_j : \chi_{jk} = 1\}$ are presumed drawn from Dirichlet distribution with parameter α_k, we set the theoretical mean θ_k to the empirical mean of S_k:

$$\theta_{km} = \langle \mathbf{p}_j \in S_k \rangle = \frac{1}{N_k} \sum_j \chi_{jk} p_{jm} \tag{27}$$

Moment matching of the m-th diagonal component of the covariance gives

$$\begin{aligned}\frac{\alpha_{km}\left(\sum_\ell \alpha_{k\ell} - \alpha_{km}\right)}{\left(\sum_\ell \alpha_{k\ell}\right)^2 \left(\sum_\ell \alpha_{k\ell} + 1\right)} =& \mathrm{cov}(S_k)_{mm} = \left\langle p_{jm}^2 \in S_k \right\rangle - \left\langle p_{jm} \in S_k \right\rangle^2 & (28)\\ \frac{\theta_{km}(1-\theta_{km})}{s_k + 1} =& \frac{1}{N_k} \sum_j \chi_{jk} p_{jm}^2 & (29)\\ s_k =& \frac{\theta_{km} - \frac{1}{N_k}\sum_j \chi_{jk} p_{jm}^2}{\frac{1}{N_k}\sum_j \chi_{jk} p_{jm}^2 - \theta_{km}^2} & (30)\end{aligned}$$

Since these M estimates s_k do not need to agree, we simply take their average.

The EM algorithm for DMM inference is summarized in Algorithm 2.

Algorithm 2 EM algorithm for Dirichlet-multinomial mixture model

```
Form pseudo-multinomial data $p_{jm} = n_{jm}/N_j$
Find $K$ clusters of this pseudodata via the $K$-means algorithm.
Initialize $\pi$ via Eq. 26
Initialize $\{\alpha_{km}\}$ via Eqs. 27 and 30
repeat
    Update $\bar{z}_{jk}$ via Eq. 21
    Update $\pi$ via Eq. 23
    repeat
        update $\{\alpha_{km}\}$ via Eq. 25
    until convergence
until convergence
```

Returning to our original task, we obtain three mixture components with Dirichlet parameters $(\alpha_{k1}, \alpha_{k2})$. The mean proportion of alt reads (WLOG we choose $m = 1$ to be alt and $m = 2$ to be ref) are $\alpha_{k1}/(\alpha_{k1} + \alpha_{k2})$, so we can assign mixture labels $k = 1, 2, 3$ to genotypes by comparing these proportions to 0 (hom ref), 1/2 (het) and 1 (hom alt). The posterior probability $\bar{z}_{jk}$ is the probability that site j has genotype k, which is exactly what we need for a probabilistic het pulldown.

C. Model-comparison test for segment merging

[SL: I'll fill this in once it's worked out.]

Appendix A: Marginalizing out latent variables of the `AllelicCapSeg`model

We wish to evaluate

$$\phi(\alpha, \beta, f, a, r) = \int_0^\infty g(\lambda, \alpha, \beta, f, a, r)\, d\lambda \tag{A1}$$

where

$$g(\lambda, \alpha, \beta, f, a, r) = \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{f_j^a (1-f)^r \lambda^{\alpha+r-1} e^{-\beta\lambda}}{(f + (1-f)\lambda)^{a+r}} \tag{A2}$$

An extremely good approximation for all values of f, α, β, and a, r is

$$g(\lambda, \alpha, \beta, f, a, r) = \frac{\lambda^{\alpha+r-1} e^{-\beta\lambda_j}}{(f + (1-f)\lambda)^{a+r}} \approx c\lambda^{\rho-1} e^{-\tau\lambda}. \tag{A3}$$

where ρ and τ are chosen to reproduce the mode of $g(\lambda, \alpha, \beta, f, a, r)$ and the curvature at its mode. Having approximated our integrand as a gamma distribution's pdf on λ, we integrate it analytically

$$\phi(\alpha, \beta, f, a, r) = c \int_0^\infty \lambda^{\rho-1} e^{-\tau\lambda}\, d\lambda = c\frac{\Gamma(\rho)}{\tau^\rho} \tag{A4}$$

The mode λ_0 is found by setting logarithmic derivatives to zero:

$$\frac{d}{d\lambda} \left[(\alpha + r - 1) \ln \lambda - \beta\lambda - n \ln (f + (1-f)\lambda)\right]_{\lambda_0} = 0 \tag{A5}$$

$$\frac{\alpha + r - 1}{\lambda_0} - \beta - \frac{n(1-f)}{f_j + (1-f_j)\lambda_0} = 0 \tag{A6}$$

Multiplying out the denominators yields a quadratic equation. Taking the positive root gives

$$\lambda_0 = \frac{\sqrt{w^2 + 4\beta f(1-f)(r + \alpha - 1} - w}{2\beta(1-f)}, \quad w = (1-f)(a - \alpha + 1) + \beta f. \tag{A7}$$

The second derivative of $\ln f$ at λ_0 is

$$\kappa = -\frac{r + \alpha - 1}{\lambda_0^2} + \frac{n(1-f)^2}{(f + (1-f)\lambda_0)^2} \tag{A8}$$

The mode of the approximating gamma distribution is $(\rho-1)/\tau$ and the log second derivative is $-(\rho-1)/\lambda_0^2$. Equating these, we obtain

$$\rho = 1 - \kappa\lambda_0^2, \quad \tau = -\kappa\lambda_0 \tag{A9}$$

Finally, we choose c so that the values of $\ln f$ and the approximation match at λ_0:

$$\ln c = \alpha \ln \beta - \ln \Gamma(\alpha) + a \ln f + r \ln(1-f) + (r + \alpha - \rho) \ln \lambda_0 + (\tau - \beta)\lambda_0 - n \ln (f + (1-f)\lambda_0) \tag{A10}$$

Algorithm 3 shows the entire computation.

Algorithm 3 Calculating $\phi(\alpha, \beta, f, a, r)$

```
$n = a + r$
$w = (1-f)(a - \alpha + 1) + \beta f$
$\lambda_0 = \left(\sqrt{w^2 + 4\beta f(1-f)(r + \alpha - 1} - w\right) / (2\beta(1-f))$
$\kappa = \left(n(1-f)^2\right) / (f + (1-f)\lambda_0)^2 - (r + \alpha - 1) / \lambda_0^2$
$\rho = 1 - \kappa\lambda_0^2$
$\tau = -\kappa\lambda_0$
$\ln c = \alpha \ln \beta - \ln \Gamma(\alpha) + a \ln f + r \ln(1-f) + (r + \alpha - \rho) \ln \lambda_0 + (\tau - \beta)\lambda_0 - n \ln (f + (1-f)\lambda_0)$
**return** $c\Gamma(\rho)/\tau^\rho$
```

Appendix B: Finding Tight Lower Bounds on Convex and Non-convex Functions

Given a function $f(x)$ and an arbitrary value x_0, we seek a lower bound $g(x) \leq f(x)$ that is tight at x_0, that is, $g(x_0) = f(x_0)$. If $f(x)$ is convex, that is, if $f'(x)$ is non-decreasing, the linearization $g(x) = f(x_0) + f'(x_0)(x - x_0)$ is such a bound. More generally, suppose $h(x)f'(x)$ is non-decreasing. Instead of approximating $f'(x) \approx f'(x_0)$, perhaps we can approximate $h(x)f'(x) \approx h(x_0)f'(x_0)$ via a candidate lower bound $g(x)$ for which

$$g'(x) = \frac{h(x_0)f'(x_0)}{h(x)} \tag{B1}$$

$$g(x) = f(x_0) + h(x_0)f'(x_0) \int_{x_0}^{x} \frac{dt}{h(t)} \tag{B2}$$

Lemma 1. *Such a function $g(x)$ is a tight lower bound on $f(x)$ if $h(x)f'(x)$ is non-decreasing for some non-negative function $h(x)$.*

Proof. By the Fundamental Theorem of Calculus,

$$f(x) - g(x) = \int_{x_0}^{x} \left(f'(t) - g'(t)\right)\, dt \tag{B3}$$

$$= \int_{x_0}^{x} \frac{h(t)f'(t) - h(x_0)f'(x_0)}{h(t)}\, dt \tag{B4}$$

By the monotonicity of $h(x)$, the integral is non-positive for $x > x_0$ and non-negative for $x < x_0$. Either way, the resulting integral is negative, so $g(x) \leq f(x)$. □

For Dirichlet-multinomial inference, we use the special case $h(x) = x$.

Corollary 1. *If $xf'(x)$ is non-decreasing for $x > 0$ then for any $x_0 > 0$*

$$f(x) \geq f(x_0) + x_0 f'(x_0)\left(\log(x) - \log(x_0)\right) \tag{B5}$$

Two important cases are $f_1(x) = \log \Gamma(x)/\Gamma(x+n)$ and $f_2(x) = \log \Gamma(x+n)/\Gamma(x) = -f_1(x)$, where n is a whole number. Using the recursive identity $\Gamma(n+1) = n\Gamma(n)$ we have

$$f_1(x) = -\sum_{k=0}^{n-1} \log(x+k), \quad f_1'(x) = -\sum_{k=0}^{n-1} \frac{1}{x+k}, \quad f_1''(x) = \sum_{k=0}^{n-1} \frac{1}{(x+k)^2} \tag{B6}$$

from which we see that $f_1(x)$ is convex and the usual linearization bound holds

$$\log \frac{\Gamma(x)}{\Gamma(x+n)} \geq \log \frac{\Gamma(x_0)}{\Gamma(x_0+n)} - \left(\psi(x_0+n) - \psi(x_0)\right)(x - x_0) \tag{B7}$$

where $\psi(x) = \frac{d}{dx} \log \Gamma(x)$ is the digamma function. As for $f_2(x) = -f_1(x)$, it is not convex, but

$$x f_2'(x) = \sum_{k=0}^{n-1} \frac{x}{x+k} = \sum_{k=0}^{n-1} \frac{1}{1+k/x}, \tag{B8}$$

which is increasing since each denominator is decreasing. Thus we apply the above corollary to obtain

$$\log \frac{\Gamma(x+n)}{\Gamma(x)} \geq \log \frac{\Gamma(x_0+n)}{\Gamma(x_0)} + x_0\left(\psi(x_0+n) - \psi(x_0)\right)\left(\log(x) - \log(x_0)\right) \tag{B9}$$