

Notes on AllelicCapSeg

Samuel K. Lee

*Broad Institute, 75 Ames Street, Cambridge, MA 02142**

(Dated: December 7, 2015)

Some notes on the methods used in the current `python` implementation and proposed methods for the `hellbender` port.

I. INTRODUCTION

[SL: A brief intro here would be nice.]

II. ReCapSeg OVERVIEW

We first summarize the portions of the ReCapSeg workflow that generate the input for AllelicCapSeg. The below is copied/paraphrased from the documentation at <http://gatkforums.broadinstitute.org/discussion/5640/recapseg-overview>.

ReCapSeg is a copy-number-variant detector that runs on user-defined target regions, which can correspond to exomes, gene panels, or arbitrary windows. ReCapSeg uses a Panel of Normal (PoN) samples to model noise and normalize the coverage calls of the target sample. These methods were designed for copy-number calling (amplification and deletion) of somatic events with a resolution of two or more targets. ReCapSeg does not need a matched normal, but operates on a panel of normal samples representing similar library preparation to agnostically remove noise. ReCapSeg is the production version of the CapSeg algorithm.

Given an DNA-Seq alignment BAM file and a BED file of target genomic intervals, the ReCapSeg algorithms estimate copy ratio by performing the following steps:

1. Generate proportional coverage: First, the per sample normalized coverage is calculated by normalizing the read coverage spanning a target segment with the total number of aligned reads (for every read group: number of reads over segment/total number of aligned reads). The proportional coverage is then calculated by normalizing every segment with the median normalized coverage across the PoN for the given segment.
2. Tangent normalization: This normalization procedure projects the sample proportional coverage to a hyperplane defined by the PoN. This normalization procedure results in a copy-ratio estimate with reduced noise.
3. Segment: The target regions are then merged into continuous segments that represent the same copy-number event. 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 Currently, ReCapSeg considers only segments that include two or more targets (a target usually represents a single exon).

To summarize, given coverage data for a set of targets, ReCapSeg produces 1) \log_2 copy-ratio estimates for each target, and 2) corresponding segments, which are specified by a set of genomic intervals. The segment files produced by ReCapSeg also contain segment “means,” which are given by the mean linear copy ratio of all targets contained within each segment.

*Electronic address: slee@broadinstitute.org

¹ Specifically, the CBS implementation provided by the R package `DNACopy` is used. Note that even though the `DNACopy` parameter `data.type` is set to `logratio`, the tangent-normalized *linear* copy ratios are used instead. [SL: Check this? If true, it is unclear to me how this affects the resulting segmentation, if at all.]

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

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

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:

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

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

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:

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

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

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

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 :

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

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 :

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

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.

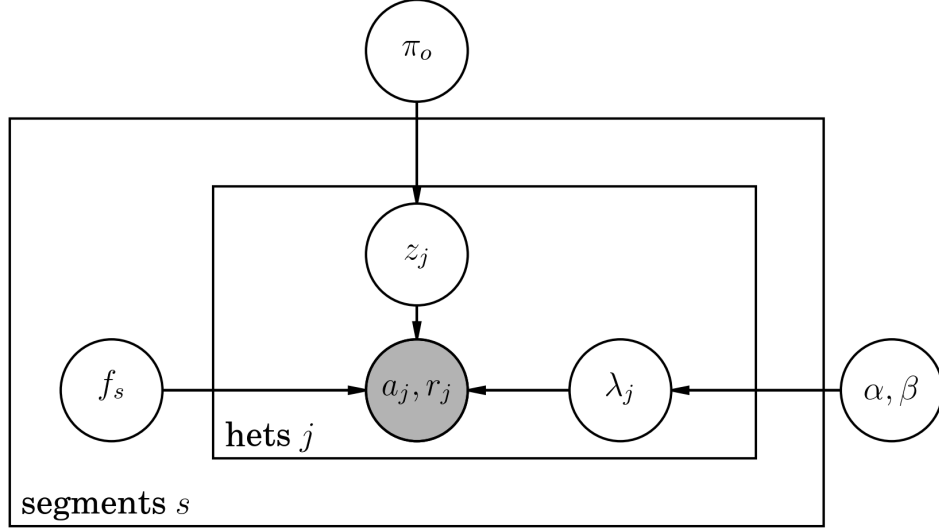


FIG. 1: Graphical model for AllelicCapSeg

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

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

The dependence on λ for alt minor sites is

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

which is messy. For ref minor sites the dependence is the same but with $f \leftrightarrow 1-f$. We 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

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

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

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

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 β , where $\alpha = \mu\beta$ (μ is the mean of $\text{Gamma}(\alpha, \beta)$). 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

```

1: for all segments  $s$  do
2:   Initialize  $f_s = \sum_{j \in s} \min(a_j, r_j) / \sum_{j \in s} n_j$ 
3: end for
4: Initialize  $\mu = 1$ 
5: Initialize  $\beta = 10$ 
6: Initialize  $\pi = 0.01$ 
7: repeat
8:   Sample each  $f_s$  with adaptive Metropolis
9:   Sample  $\pi$  with adaptive Metropolis
10:  Sample  $\mu$  with adaptive Metropolis
11:  Sample  $\beta$  with adaptive Metropolis
12: until convergence

```

B. Target/SNP segment union

[SL: DB can fill this in.]

C. 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 i th 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 $|\tau_i - \tau_{i \pm 1}| + |f_i - f_{i \pm 1}|$ 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.

D. Parameter optimization
E. Similar-segment merging
F. Final parameter optimization
IV. PROPOSED METHODS FOR hellbender
A. 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 it 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

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 .

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 $\mathbf{z}_j = (0, 0, \dots, 1, 0, 0, \dots)$, 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 :

$$P(\mathbf{n}_j | z_{jk} = 1, \alpha_k) = \frac{\Gamma(A_k)}{\Gamma(A_k + N_j)} \prod_m \frac{\Gamma(\alpha_{km} + n_{jm})}{\Gamma(\alpha_{km})}, \quad (12)$$

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 the DMM is

$$P(\mathbf{z}, \mathbf{n} | \pi, \alpha) = P(\mathbf{z} | \pi) P(\mathbf{n} | \mathbf{z}, \alpha) \quad (13)$$

$$= \prod_{jk} \left[\pi_k \frac{\Gamma(A_k)}{\Gamma(A_k + N_j)} \prod_m \frac{\Gamma(\alpha_{km} + n_{jm})}{\Gamma(\alpha_{km})} \right]^{z_{jk}} \quad (14)$$

In the E step of the EM algorithm, we obtain the posterior distribution on $P(\mathbf{z} | \mathbf{n}, \pi, \alpha)$ from Eq. (14). By inspection the posterior is a product of independent multinomials

$$\bar{z}_{jk} \equiv P(z_{jk} = 1 | \mathbf{n}, \pi, \alpha) \propto \pi_k \frac{\Gamma(A_k)}{\Gamma(A_k + N_j)} \prod_m \frac{\Gamma(\alpha_{km} + n_{jm})}{\Gamma(\alpha_{km})}, \quad (15)$$

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

$$\sum_{jk} \bar{z}_{jk} \left\{ \log \pi_k + \log \frac{\Gamma(A_k)}{\Gamma(A_k + N_j)} + \sum_m \log \frac{\Gamma(\alpha_{km} + n_{jm})}{\Gamma(\alpha_{km})} \right\}. \quad (16)$$

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

$$\pi_k = \frac{\sum_j \bar{z}_{jk}}{\sum_{j\ell} \bar{z}_{j\ell}} \quad (17)$$

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 (C7) and (C9) and dropping additive constants, we find that the iterative step is to maximize the lower bound

$$\sum_{jk} \bar{z}_{jk} \left\{ - \left(\psi(\hat{A}_k + N_j) - \psi(\hat{A}_k) \right) A_k + \sum_m \hat{\alpha}_{km} \left(\psi(\hat{\alpha}_{km} + n_{jm}) - \psi(\hat{\alpha}_{km}) \right) \log(\alpha_{km}) \right\}. \quad (18)$$

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

$$\alpha_{km} = \hat{\alpha}_{km} \frac{\sum_j \bar{z}_{jk} (\psi(\hat{\alpha}_{km} + n_{jm}) - \psi(\hat{\alpha}_{km}))}{\sum_j \bar{z}_{jk} (\psi(\hat{A}_k + N_j) - \psi(\hat{A}_k))} \quad (19)$$

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

$$\pi_k = \frac{\sum_j \chi_{jk}}{N} = \frac{N_k}{N} \quad (20)$$

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 :

$$\theta_{km} = \langle \mathbf{p}_j \in S_k \rangle = \frac{1}{N_k} \sum_j \chi_{jk} p_{jm} \quad (21)$$

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

$$\frac{\alpha_{km} (\sum_\ell \alpha_{k\ell} - \alpha_{km})}{(\sum_\ell \alpha_{k\ell})^2 (\sum_\ell \alpha_{k\ell} + 1)} = \text{cov}(S_k)_{mm} = \langle p_{jm}^2 \in S_k \rangle - \langle p_{jm} \in S_k \rangle^2 \quad (22)$$

$$\frac{\theta_{km}(1 - \theta_{km})}{s_k + 1} = \frac{1}{N_k} \sum_j \chi_{jk} p_{jm}^2 \quad (23)$$

$$s_k = \frac{\theta_{km} - \frac{1}{N_k} \sum_j \chi_{jk} p_{jm}^2}{\frac{1}{N_k} \sum_j \chi_{jk} p_{jm}^2 - \theta_{km}^2} \quad (24)$$

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

- 1: Form pseudo-multinomial data $p_{jm} = n_{jm}/N_j$
 - 2: Find K clusters of this pseudodata via the K -means algorithm.
 - 3: Initialize π via Eq. 20
 - 4: Initialize $\{\alpha_{km}\}$ via Eqs. 21 and 24
 - 5: **repeat**
 - 6: Update \bar{z}_{jk} via Eq. 15
 - 7: Update π via Eq. 17
 - 8: **repeat**
 - 9: update $\{\alpha_{km}\}$ via Eq. 19
 - 10: **until** convergence
 - 11: **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.

B. 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 AllelicCapSegmodel

We wish to evaluate

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

where

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Algorithm 3 shows the entire computation.

Appendix B: Initializing the AllelicCapSeg Model

A very reasonable way to initialize the model would be to find the mode of the likelihood via the EM algorithm, where in the E step we compute expectations of the posterior on latent variables z and λ and in the M step we maximize the log likelihood averaged over the E step posterior. We know the E step is feasible because it involves the same computations as marginalizing over the latent variables, which we accomplished above. Although the M step is intractable, we can perform a poor-man's EM in which we employ a heuristically reasonable M step. This will suffice to significantly reduce burn-in time of our MCMC sampling.

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

```

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

```

The first E step quantities required are the responsibilities for each locus to be alt minor, ref minor, or outlier, that is, the posteriors $\bar{z}_{ja} = P(z_{ja} = 1)$ etc. These are easily read off from Equation 10, which contains the unnormalized posteriors.

$$(\bar{z}_{ja}, \bar{z}_{jr}, \bar{z}_{jo}) \propto \left((1 - \pi)\phi(\alpha, \beta, f_j, a_j, r_j), (1 - \pi)\phi(\alpha, \beta, 1 - f_j, a_j, r_j), \frac{2\pi a_j! r_j!}{(n_j + 1)!} \frac{\Gamma(\alpha)}{\beta^\alpha} \right) \quad (\text{B1})$$

To initialize minor allele fractions set f_S to the responsibility-weighted number of minor read alleles divided by the total number of reads in segment S :

$$f_S = \frac{\sum_{j \in S} \bar{z}_{ja} a_j + \bar{z}_{jr} r_j}{\sum_{j \in S} \bar{z}_{ja} + \bar{z}_{jr}} \quad (\text{B2})$$

In fact, it is easily shown that this is the exact M step when $\lambda_j = 1$ for all j . Similarly, we initialize π to the responsibility-weighted proportion of outliers

$$\pi = \frac{1}{N} \sum_j \bar{z}_{jo} \quad (\text{B3})$$

This, is, in fact, the exact M step for π . Our initialization of μ and β is somewhat more heuristic but is based on moment matching and is probably asymptotically exact as $N \rightarrow \infty$. We set μ to the responsibility-weighted posterior sample mean of $\{\lambda\}$ and μ/β to the sample variance. Since the outlier distribution is determined by μ and β alone we exclude outliers in order to make our initialization depend more on the data and less on our initial guess. Thus we have $\mu = \langle \lambda \rangle$, $\mu/\beta = \langle \lambda^2 \rangle - \langle \lambda \rangle^2$, where

$$\langle \lambda^k \rangle = \frac{\sum_j \left(\bar{z}_{ja} \langle \lambda_j^k \rangle_a + \bar{z}_{jr} \langle \lambda_j^k \rangle_r \right)}{\sum_j (\bar{z}_{ja} + \bar{z}_{jr})} \quad (\text{B4})$$

and $\langle \lambda_j^k \rangle_{a(r)}$ is the posterior expectation of λ_j^k given $z_{ja(r)} = 1$. These can be calculated quite easily using our previous work. Recall that the unnormalized conditional distribution on λ_j in the alt minor case was

$$g(\lambda_j, \alpha, \beta, f_j, a_j, r_j) = \frac{f_j^{a_j} (1 - f_j)^{r_j} \lambda_j^{\alpha + r_j - 1} e^{-\beta \lambda_j}}{(f_j + (1 - f_j)\lambda_j)^{n_j}}, \quad (\text{B5})$$

and that we can efficiently compute $\phi(\alpha, \beta, f_j, a_j, r_j) = \int_0^\infty g(\lambda_j, \alpha, \beta, f_j, a_j, r_j) d\lambda$. The posterior expectation of λ_j^k is therefore

$$\langle \lambda_j^k \rangle_a = \frac{\int_0^\infty \lambda^k g(\lambda, \alpha, \beta, f, a, r) d\lambda}{\int_0^\infty g(\lambda, \alpha, \beta, f, a, r) d\lambda} = \frac{\int_0^\infty \lambda^k g(\lambda, \alpha + k, \beta, f, a, r) d\lambda}{\int_0^\infty g(\lambda, \alpha, \beta, f, a, r) d\lambda} = \frac{\phi(\alpha + k, \beta, f, a, r)}{\phi(\alpha, \beta, f, a, r)} \quad (\text{B6})$$

As before, the ref minor case is obtained from $f \leftrightarrow 1 - f$. We present the complete initialization process in Algorithm 4.

Appendix C: 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

Algorithm 4 Initializing the AllelicCapSeg Model

```

1: Set  $\bar{z}_{ja} = 1$  if  $a_j < r_j$ , else  $\bar{z}_{jr} = 1$ .
2: INITIALIZEALLELEFRACTION
3: Set  $\pi = 0.01$ 
4: Set  $\mu = 1$ 
5: Set  $\beta = 10$ 

6: repeat
7:   CALCULATERESPONSIBILITIES
8:   INITIALIZEALLELEFRACTION
9:   INITIALIZEOUTLIERPROBABILITY
10:  INITIALIZEGAMMAHYPERPARAMETERS
11: until a low standard of convergence

12: procedure CALCULATERESPONSIBILITIES
13:    $(p_{ja}, p_{jr}, p_{jo}) = \left( (1 - \pi)\phi(\alpha, \beta, f_j, a_j, r_j), (1 - \pi)\phi(\alpha, \beta, 1 - f_j, a_j, r_j), \frac{2\pi a_j! r_j!}{(n_j + 1)!} \frac{\Gamma(\alpha)}{\beta^\alpha} \right)$  for each  $j$ .
14:    $(\bar{z}_{ja}, \bar{z}_{jr}, \bar{z}_{jo}) = (p_{ja}, p_{jr}, r_{jo}) / (p_{ja} + p_{jr} + p_{jo})$  for each  $j$ 
15: end procedure

16: procedure INITIALIZEALLELEFRACTION
17:   Set  $f_S = \sum_{j \in S} (\bar{z}_{ja} a_j + \bar{z}_{jr} r_j) / \sum_{j \in S} (\bar{z}_{ja} + \bar{z}_{jr})$  for each  $S$ 
18: end procedure

19: procedure INITIALIZEOUTLIERPROBABILITY
20:   Set  $\pi = \sum_j \bar{z}_{jo} / N$ 
21: end procedure

22: procedure INITIALIZEGAMMAHYPERPARAMETERS
23:    $\langle \lambda_j^k \rangle_a = \phi(\alpha + k, \beta, f_j, a_j, r_j) / \phi(\alpha, \beta, f, a, r)$  for  $k = 1, 2$  for all  $j$ 
24:    $\langle \lambda_j^k \rangle_r = \phi(\alpha + k, \beta, 1 - f_j, a_j, r_j) / \phi(\alpha, \beta, 1 - f_j, a_j, r_j)$  for  $k = 1, 2$  for all  $j$ 
25:    $\langle \lambda^k \rangle = \sum_j (\bar{z}_{ja} \langle \lambda_j^k \rangle_a + \bar{z}_{jr} \langle \lambda_j^k \rangle_r) / \sum_j (\bar{z}_{ja} + \bar{z}_{jr})$  for  $k = 1, 2$ 
26:   Set  $\mu = \langle \lambda \rangle$ 
27:   Set  $\beta = \langle \lambda \rangle / (\langle \lambda^2 \rangle - \langle \lambda \rangle^2)$ 
28: end procedure

```

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

$$g'(x) = \frac{h(x_0)f'(x_0)}{h(x)} \quad (\text{C1})$$

$$g(x) = f(x_0) + h(x_0)f'(x_0) \int_{x_0}^x \frac{dt}{h(t)} \quad (\text{C2})$$

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,

$$f(x) - g(x) = \int_{x_0}^x (f'(t) - g'(t)) dt \quad (\text{C3})$$

$$= \int_{x_0}^x \frac{h(t)f'(t) - h(x_0)f'(x_0)}{h(t)} dt \quad (\text{C4})$$

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 $g(x) \leq f(x)$. \square

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 $x_0 > 0$*

$$f(x) \geq f(x_0) + x_0 f'(x_0) (\log(x) - \log(x_0)) \quad (\text{C5})$$

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

$$f_1(x) = -\sum_{k=0}^{n-1} \log(x+k), \quad f_1'(x) = -\sum_{k=0}^{n-1} \frac{1}{x+k}, \quad f_1''(x) = \sum_{k=0}^{n-1} \frac{1}{(x+k)^2} \quad (\text{C6})$$

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

$$\log \frac{\Gamma(x)}{\Gamma(x+n)} \geq \log \frac{\Gamma(x_0)}{\Gamma(x_0+n)} - (\psi(x_0+n) - \psi(x_0)) (x - x_0) \quad (\text{C7})$$

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

$$xf_2'(x) = \sum_{k=0}^{n-1} \frac{x}{x+k} = \sum_{k=0}^{n-1} \frac{1}{1+k/x}, \quad (\text{C8})$$

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

$$\log \frac{\Gamma(x+n)}{\Gamma(x)} \geq \log \frac{\Gamma(x_0+n)}{\Gamma(x_0)} + x_0 (\psi(x_0+n) - \psi(x_0)) (\log(x) - \log(x_0)) \quad (\text{C9})$$