Notes on AllelicCapSeg

Samuel K. Lee
Broad Institute, 75 Ames Street, Cambridge, MA 02142[*]
(Dated: September 2, 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 are 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 $\log_2$ of 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 AllelicCapSeg WORKFLOW

A. Segmented model

AllelicCapSeg uses allele-count data from heterozygous SNP sites to improve upon the segmented copy-number model given by ReCapSeg. The AllelicCapSeg model is characterized by a set of segment intervals and a set of model parameters; a model with N segments is taken to have $2N + 2$ parameters

$$\{\tau_1, f_1, \ldots \tau_N, f_N, \sigma_g, \gamma\} \,. \tag{1}$$

Each individual segment labeled by i is specified by its true copy ratio τ_i and its true minor allele fraction f_i (referred to as the "minor homologous chromosome fraction" in the python code), yielding $2N$ parameters. Two additional global parameters σ_g and γ determine the variance of the copy ratio and the allele-fraction bias (referred to as the "skew"), respectively, for all segments.

Specifically, the variance of the copy ratio for targets in the ith segment is assumed to be given by $\sigma_i = \sigma_g \tau_i$. Furthermore, the allele-fraction bias γ attempts to correct for a global discrepancy between the measured and true allele fractions, which may be induced by effects such as reference bias during alignment. It is defined such that a balanced true allele fraction of $f = 1/2$ will yield, on average, an observed minor allele fraction of $\gamma/2$; we shall expand on this definition below.

B. Identifying normal heterozygous SNP sites (het pulldown)

The main idea behind AllelicCapSeg is to use allele-count data at heterozygous SNP sites to infer copy number; these sites should exhibit balanced reference/alternate read counts in the normal sample, but unbalanced copy-number events could measurably alter this balance in the tumor sample. Thus, the first step in the AllelicCapSeg workflow is to identify the heterozygous SNP sites that will be leveraged in the rest of the analysis.

We first start with a list of common SNP sites, and consider all reads passing a quality-score filter (taking a default minimum quality of 0). We examine the allele counts at these sites in the normal sample, and discard those sites that appear to be homozygous by performing a two-tailed binomial test on the counts at each site.

Specifically, for a SNP site labeled by k, let the reference and alternate counts be A_k and B_k, respectively; also, let the major and minor allele counts be M_k and m_k, respectively. The total counts are denoted by $N_k = A_k + B_k = M_k + m_k$.[2]

For sites above a read-count threshold $N_k \geq 10$, the compatibility of the major allele count M_k and the total count N_k with the null hypothesis of a allele fraction f (assumed to be the same across all sites, with a default value of $f = 1/2$ used) is checked with a two-tailed binomial test; a p-value threshold of 0.05 is assumed. Sites with counts below the threshold or incompatible with the allele fraction f (i.e., likely to be homozygous SNPs) are filtered out.

The remaining sites are assumed to be heterozygous SNPs. The allele counts $\{A_k, B_k\}$ at these sites are pulled from the tumor sample and stored.

C. Initializing the model

The initial state of the model is based on the result from ReCapSeg. The set of segment intervals is set to those found by ReCapSeg, and the τ_i are set to the median copy ratio of all targets contained within each ith segment (note that these differ from the aforementioned segment "means" returned by ReCapSeg, which are not used by AllelicCapSeg). Initial values of $\sigma_g = 0.15$ and $\gamma = 2 \times 0.48 = 0.96$ are used for the global parameters.

Initialization of the minor homologous chromosome fractions f_i for each segment is more involved. This is accomplished on a per segment basis by taking f_i to be the single parameter in a beta-binomial mixture model and selecting the value that maximizes the corresponding likelihood function. Recall that the beta-binomial distribution gives the probability for the number of successes k out of a number of trials n, where the probability of success for each trial is a random variable that follows a beta distribution with parameters α and β. The probability distribution is

$$p_{\beta\text{-bin}}(k|n, \alpha, \beta) = \binom{n}{k} \frac{B(k+\alpha, n-k+\beta)}{B(\alpha, \beta)} \,, \tag{2}$$

[2] Note that the "alternate" and "minor" counts are simply found by subtracting the reference and major counts from the total count, respectively; this lumps stray reads that do not actually match the alternate or minor alleles into both of these categories. In principle, we could easily take only the largest and second-largest counts and discard the rest, but this is not what is currently implemented.

where $B(\alpha, \beta)$ is the Euler beta function.

Consider a segment labeled by i, and let $D_i = \{A_{k_i}, B_{k_i}, \ldots, A_{k_i+K_i}, B_{k_i+K_i}\}$ be the data set of tumor-sample allele counts for the K_i heterozygous SNP sites (i.e., those identified in the het pulldown step) contained within this segment. The likelihood function for the allele fraction f_i of this segment, given the data set D_i, is taken to be

$$\mathcal{L}(f_i|D_i) = p(D_i|f_i) = \begin{cases} 1\,, & D_i = \{\}\,, \\ \displaystyle\prod_{k=k_i}^{k_i+K_i} \left\{\frac{1}{2}(1-w)\left[\mathcal{L}_{\mathrm{ref}}(f_i|A_k, B_k) + \mathcal{L}_{\mathrm{alt}}(f_i|A_k, B_k)\right] + w\mathcal{L}_{\mathrm{out}}\right\}\,, & \text{otherwise}\,. \end{cases} \tag{3}$$

Here, w is a weight parameter that will be later be used to account for outliers via the addition of a uniform-distribution component to the mixture model, which is represented in the likelihood by the $\mathcal{L}_{\mathrm{out}}$ term; for this initialization stage, $w = 0$ is chosen.[3]

Furthermore, $\mathcal{L}_{\mathrm{ref}}(f_i|A_k, B_k)$ is the likelihood of the minor allele fraction f_i, given allele counts $\{A_k, B_k\}$ at SNP site k, for the case of a reference minor allele (i.e., $A_k \leq B_k$). It is given by

$$\mathcal{L}_{\mathrm{ref}}(f_i|A_k, B_k) = \begin{cases} 1\,, & f_i = 0 \text{ and } A_i = 0\,, \\ p_{\beta\text{-bin}}(B_k|A_k + B_k, \alpha_{\mathrm{ref}}(f_i), \beta_{\mathrm{ref}}(f_i))\,, & \text{otherwise}\,. \end{cases} \tag{4}$$

Here,

$$\alpha_{\mathrm{ref}}(f_i) = \left(1 - \frac{f_i}{\gamma}\right)\Sigma\,, \qquad \beta_{\mathrm{ref}}(f_i) = \frac{f_i \Sigma}{\gamma}\,, \tag{5}$$

where the parameter Σ accounts for overdispersion and is fit to a panel of normal samples, yielding

$$\Sigma = \exp(s_0 + s_1\gamma/2)\,, \qquad s_0 = -33.67232\,, \quad s_1 = 82.77464\,. \tag{6}$$

Similarly, $\mathcal{L}_{\mathrm{alt}}(f_i|A_k, B_k)$ is the likelihood of f_i, given allele counts $\{A_k, B_k\}$ at SNP site k, for the case of an alternate minor allele (i.e., $B_k \leq A_k$). It is given by

$$\mathcal{L}_{\mathrm{alt}}(f_i|A_k, B_k) = p_{\beta\text{-bin}}(B_k|A_k + B_k, \alpha_{\mathrm{alt}}(f_i), \beta_{\mathrm{alt}}(f_i))\,, \tag{7}$$

with

$$\alpha_{\mathrm{alt}}(f_i) = f_i\gamma\Sigma\,, \qquad \beta_{\mathrm{alt}}(f_i) = (1 - f_i\gamma)\,\Sigma\,. \tag{8}$$

[SL: I'm not sure I fully understand the forms of α_{ref}, β_{ref}, α_{alt}, and β_{alt} yet, but I probably just need to sit down with these equations for a bit. Will add explanations of these terms and γ here once I do. Also, it would be nice to double check that the overdispersion fit is reasonable for the relevant PoNs; would be even better if we can get rid of this fit!]

In practice, the maximum-likelihood estimate for f_i is found by using the `scipy` implementation of the L-BFGS-B optimization algorithm to minimize the negative log-likelihood function, requiring convergence to machine precision. An initial guess of $f_i = 0.25$ is used, with the allowed range taken to be $10^{-10} \leq f_i \leq 0.5$. [SL: It's not clear to me what value of f_i is actually returned for the case where the segment does not contain any SNPs, $D_i = \{\}$. It might just be the initial guess of $f_i = 0.25$, but we can check this. If this is the case, we should check if initialization of meaningless values of f_i for such segments affects things downstream.]

[3] [SL: Although, strictly speaking, $w = 0$ should be used for initialization, the same value of $w = 0.0005$ used in the beta-binomial–uniform mixture model is used instead. However, this does not have any effect on the resulting initial values of f_i, since the weight factor just contributes to an overall multiplicative constant in the likelihood. Unfortunately, a further complication is that I think the likelihood for the beta-binomial–uniform mixture model is also incorrectly constructed in code; schematically, the quantity appearing under the product is coded as $1/2(1-w)(\mathcal{L}_{\mathrm{ref}} + \mathcal{L}_{\mathrm{alt}}) + w'\mathcal{L}_{\mathrm{out}}$, with $w' = 0.005 \neq w$. Finally, the implementation essentially sets $\mathcal{L}_{\mathrm{out}} = 1$ (albeit in a very roundabout manner). I'm not sure if this is the desired behavior, and it also might not be consistent with how targets that are outliers in copy ratio are treated.]

D. Initial optimization of allelic-fraction bias

E. SNP segmentation

Using the initial value of γ found in the previous step, SNPs and their corresponding reference/alternate read counts are transformed to targets with corresponding effective "copy ratios," which are then segmented using the `DNACopy` CBS implementation. However, these effective copy ratios will not scale with the copy number as would a true copy ratio, and hence should not be thought of as such.

In particular, for a SNP site indexed by k with allele counts $\{A_k, B_k\}$, the effective copy ratio $\tilde{\tau}_k$ is taken to be

$$\tilde{\tau}_k = \left| \frac{B_k}{A_k + B_k} - \frac{\gamma}{2} \right| . \tag{9}$$

The idea here is that the quantity $B_k/(A_k + B_k) - \gamma/2$ will be clustered around zero for sites k contained in segments i where the true minor allele fraction is $f_i = 1/2$, but will generally form two clusters centered at $\pm$ [SL: fill in once I understand what is going on with γ], with the positive and negative signs corresponding to sites with reference and alternate minor alleles, respectively. By taking the effective copy ratio passed to CBS to be the absolute value of this quantity, we are simply folding the data vertically before segmenting. [SL: Does this somehow screw up the statistics of the balanced segments by artificially reducing their variance? Could this have an effect on CBS?] As with the segmentation of targets in `ReCapSeg`, the `DNACopy` parameter `data.type` is set to `logratio`. [SL: Actually, I think that $2^{\tilde{\tau}_k}$ is being passed to `DNACopy`, not just $\tilde{\tau}_k$. Again, I'm not sure if this actually affects the segmentation, but it doesn't seem consistent with how we treat targets. I think we should try segmenting a quantity that is proportional to copy ratio in either linear or log space for both targets and SNPs.]

F. Target/SNP segment union

[SL: DB can fill this in.]

G. Small-segment merging

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.[4] 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.

[4] 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.

H. Parameter optimization

I. Similar-segment merging

J. 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 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 $P(\mathbf{z}_j) = \mathrm{Mult}(\mathbf{z}_j|\pi)$, 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:

$$P(\mathbf{n}_j \mid z_{jk} = 1, \alpha_k) = \frac{\Gamma(A_k)}{\Gamma(A_k + N_j)} \prod_m \frac{\Gamma(\alpha_{km} + n_{jm})}{\Gamma(\alpha_{km})}, \tag{10}$$

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

$$P(\mathbf{z}, \mathbf{n} \mid \pi, \alpha) = P(\mathbf{z} \mid \pi) P(\mathbf{n} \mid \mathbf{z}, \alpha) \tag{11}$$

$$= \prod_{jk} \left[\pi_k \frac{\Gamma(A_k)}{\Gamma(A_k + N_j)} \prod_m \frac{\Gamma(\alpha_{km} + n_{jm})}{\Gamma(\alpha_{km})} \right]^{z_{jk}} \tag{12}$$

In the E step of the EM algorithm, we obtain the posterior distribution on $P(\mathbf{z} \mid \mathbf{n}, \pi, \alpha)$ from Eq. (12). By inspection the posterior is a product of independent multinomials

$$\bar{z}_{jk} \equiv P(z_{jk} = 1 \mid \mathbf{n}, \pi, \alpha) \propto \pi_k \frac{\Gamma(A_k)}{\Gamma(A_k + N_j)} \prod_m \frac{\Gamma(\alpha_{km} + n_{jm})}{\Gamma(\alpha_{km})}, \tag{13}$$

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

$$\sum_{jk} \bar{z}_{jk} \left\{ \log \pi_k + \log \frac{\Gamma(A_k)}{\Gamma(A_k + N_j)} + \sum_m \log \frac{\Gamma(\alpha_{km} + n_{jm})}{\Gamma(\alpha_{km})} \right\}. \tag{14}$$

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

$$\pi_k = \frac{\sum_j \bar{z}_{jk}}{\sum_{j\ell} \bar{z}_{j\ell}} \tag{15}$$

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 (A7) and (A9) and dropping additive constants, we find that the iterative step is to maximize the lower bound

$$\sum_{jk} \bar{z}_{jk} \left\{ -\left(\psi(\hat{A}_k + N_j) - \psi(\hat{A}_k)\right) A_k + \sum_m \hat{\alpha}_{km} \left(\psi(\hat{\alpha}_{km} + n_{jm}) - \psi(\hat{\alpha}_{km})\right) \log(\alpha_{km}) \right\}. \tag{16}$$

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

$$\alpha_{km} = \hat{\alpha}_{km} \frac{\sum_j \bar{z}_{jk} \left(\psi(\hat{\alpha}_{km} + n_{jm}) - \psi(\hat{\alpha}_{km})\right)}{\sum_j \bar{z}_{jk} \left(\psi(\hat{A}_k + N_j) - \psi(\hat{A}_k)\right)} \tag{17}$$

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

$$\pi_k = \frac{\sum_j \chi_{jk}}{N} = \frac{N_k}{N} \tag{18}$$

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:

$$\theta_{km} = \langle \mathbf{p}_j \in S_k \rangle = \frac{1}{N_k} \sum_j \chi_{jk} p_{jm} \tag{19}$$

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

$$\frac{\alpha_{km}\left(\sum_\ell \alpha_{k\ell} - \alpha_{km}\right)}{\left(\sum_\ell \alpha_{k\ell}\right)^2 \left(\sum_\ell \alpha_{k\ell} + 1\right)} = \mathrm{cov}(S_k)_{mm} = \left\langle p_{jm}^2 \in S_k \right\rangle - \left\langle p_{jm} \in S_k \right\rangle^2 \tag{20}$$

$$\frac{\theta_{km}(1 - \theta_{km})}{s_k + 1} = \frac{1}{N_k} \sum_j \chi_{jk} p_{jm}^2 \tag{21}$$

$$s_k = \frac{\theta_{km} - \frac{1}{N_k} \sum_j \chi_{jk} p_{jm}^2}{\frac{1}{N_k} \sum_j \chi_{jk} p_{jm}^2 - \theta_{km}^2} \tag{22}$$

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

The EM algorithm for DDM inference is summarized in Algorithm 1.

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.

[DB: We still need to figure out how to use the beta-binomial distribution on het alt and ref counts in `AllelicCapSeg`.]

Algorithm 1 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 $\pi$ via Eq. 18
4: Initialize $\{\alpha_{km}\}$ via Eqs. 19 and 22
5: repeat
6:     Update $\bar{z}_{jk}$ via Eq. 13
7:     Update $\pi$ via Eq. 15
8:     repeat
9:         update $\{\alpha_{km}\}$ via Eq. 17
10:    until convergence
11: until convergence
```

B. Model-comparison test for segment merging

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

Appendix A: 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

$$g'(x) = \frac{h(x_0)f'(x_0)}{h(x)} \tag{A1}$$

$$g(x) = f(x_0) + h(x_0)f'(x_0)\int_{x_0}^{x} \frac{dt}{h(t)} \tag{A2}$$

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,

$$f(x) - g(x) = \int_{x_0}^{x} \left(f'(t) - g'(t)\right)\, dt \tag{A3}$$

$$= \int_{x_0}^{x} \frac{h(t)f'(t) - h(x_0)f'(x_0)}{h(t)}\, dt \tag{A4}$$

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 $x_0 > 0$*

$$f(x) \geq f(x_0) + x_0 f'(x_0)\left(\log(x) - \log(x_0)\right) \tag{A5}$$

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

$$f_1(x) = -\sum_{k=0}^{n-1}\log(x+k), \quad f_1'(x) = -\sum_{k=0}^{n-1}\frac{1}{x+k}, \quad f_1''(x) = \sum_{k=0}^{n-1}\frac{1}{(x+k)^2} \tag{A6}$$

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

$$\log \frac{\Gamma(x)}{\Gamma(x+n)} \geq \log \frac{\Gamma(x_0)}{\Gamma(x_0+n)} - \left(\psi(x_0+n) - \psi(x_0)\right)(x - x_0) \tag{A7}$$

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

$$x f_2'(x) = \sum_{k=0}^{n-1} \frac{x}{x+k} = \sum_{k=0}^{n-1} \frac{1}{1 + k/x}, \tag{A8}$$

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

$$\log \frac{\Gamma(x+n)}{\Gamma(x)} \geq \log \frac{\Gamma(x_0+n)}{\Gamma(x_0)} + x_0 \left(\psi(x_0+n) - \psi(x_0)\right)\left(\log(x) - \log(x_0)\right) \tag{A9}$$