Homework 7: SNV Genotyping

GENOME 541: Cancer Genomics Module (Spring 2023)

Instructor: Gavin Ha

Due May 19, 2023 @ 11:59pm

You will implement a single-nucleotide variant (SNV) caller. You will write functions for equations described in Lecture 2 for a standard, single-sample binomial mixture model. For each function that you implement, you will use it to generate results for the initial settings and the first iteration of the EM algorithm. Then, you will implement the full EM algorithm to predict the mutation status for the input data.

The input data is a text file `Homework7_alleleCounts.txt` that contains a set of 1000 loci with reference and non-reference allele counts. The outputs for each function are provided so that you can use it to check your code.

Overall, your tasks are to implement the mixture model functions, implement and run the EM algorithm to convergence, and then annotate the mutation status for each locus. This assignment is divided into 4 parts:

0. Initial setup of the libraries and data
1. Implement the binomial likelihood model
2. Implement functions for the EM algorithm
3. Implement and run the full EM algorithm to learn the parameters and infer the genotypes.

Code template files `Homework7_SNVGenotyping_template.Rmd` and `Homework7_SNVGenotyping_template.ipynb` are provided for you to write your code and to generate outputs. This assignment instruction was generated using R Markdown, so the code syntax will be specific to R. However, the Python Jupyter notebook template contains the equivalent code in Python.

Please submit the following files:

1. The code as an R Markdown (`Homework7_code_FirstName-LastName.Rmd`) or Python Jupyter Notebook (`Homework7_code_FirstName-LastName.ipynb`) file.
2. The exported PDF (`Homework7_code_FirstName-LastName.pdf`) of the code and output.
3. The final output file of the results in Section 3.1 (`Homework7_mutationCalls_FirstName-LastName.txt`)

Total Points for Assignment: 60

0. Setup the libraries and input data

Total Points: 5

0.1. Load libraries

```
library(ggplot2) # for plotting in Section 3
```

0.2. Load the input data

The file containing allelic counts for T loci. Each locus i contains the reference base (A, `refCount`) count (x_i) and non-reference/variant (B, `NrefCount`) base count. The read depth (N_i, `depth`) at each locus i is the sum of the counts for both bases $A + B$ at that locus.

```r
counts <- read.delim("Homework7_alleleCounts.txt", as.is = TRUE)
counts[1:2, ]
```

```
##   chr position refBase refCount NRefBase NrefCount depth
## 1   2    91822       C       42        N         2    44
## 2   2   202466       C       36        N        12    48
```

```r
x <- counts$refCount
N <- counts$depth
```

0.3. Initialize model parameters and hyperparameters.

Initialize parameters and hyperparameters for genotypes, AA, AB, BB,

- `mu.init`, $\mu_{1:K}^{(0)}$ for the binomial parameter
- `pi.init`, $\pi_{1:K}^{(0)}$ for the genotype probability (mixed weights)
- `alpha.hyper`, $\alpha_{1:K}$ for the Beta prior
- `beta.hyper`, $\beta_{1:K}$ for the Beta prior
- `delta.hyper`, $\delta_{1:K}$ for the Dirichlet prior

```r
mu.init <- c(0.99, 0.5, 0.01)
pi.init <- c(0.80, 0.15, 0.05)
alpha.hyper <- c(10, 5, 1)
beta.hyper <- c(1, 5, 10)
delta.hyper <- c(8, 2, 2)
```

1. Implement functions for the Binomial Mixture Model

Total Points for Section: 10

1.1. Compute the observed binomial likelihood probabilities (Points: 5)

1.1.1. Define a function, `compute.binom.lik` This function will compute the binomial probability for the input x_i at each locus i and (conditioned on) each genotype $k \in \{AA, AB, BB\}$.

$$p(x_i|Z_i = k, N_i) = Bin(x_i|N_i, \mu_k)$$

The function should take as arguments:

- `x`, the input reference base counts $x_{1:T}$
- `N`, the input depth $N_{1:T}$
- `mu`, the binomial parameters $\mu_{1:K}$

The function should return a matrix of binomial probabilities with dimensions $T \times K$.

You can use the built-in `dbinom` function for the binomial probability mass function (pmf).

```
compute.binom.lik <- function(x, N, mu){

}
```

1.1.2. Compute the binomial probabilities Use the `compute.binom.lik` function you wrote in Section 1.1.1 to compute the binomial probabilities for $x_{1:T}$ and $N_{1:T}$ from the input file. Use initial binomial parameters set in Section 0.3, `mu.init`.

Save this to a variable named `obs.lik`. `obs.lik` should have dimensions $T \times K$ (i.e. 1000×3). Print the first 5 lines of `obs.lik`.

Here is the call to the function and the output for you to check your code.

```
obs.lik <- compute.binom.lik(x, N, mu.init)
obs.lik[1:5, ]
```

```
##                AA           AB           BB
## [1,] 6.202536e-02 5.377387e-11 9.271746e-82
## [2,] 4.851809e-14 2.475124e-04 6.175313e-62
## [3,] 8.255780e-03 1.403009e-09 1.197446e-76
## [4,] 6.978069e-07 2.109893e-04 8.531618e-47
## [5,] 4.505405e-09 1.119110e-04 5.677071e-55
```

1.2. Compute the log likelihood function ℓ (Points: 5)

1.2.1. Define a function, `compute.loglik`, that will compute the log likelihood function for the current parameter setttings

$$\ell = \sum_{i=1}^{T} \log \left(\sum_{k=1}^{K} \pi_k Bin(x_i|N_i, \mu_k) \right)$$

The function should take as arguments:

- `obs.lik`, the likelihood (binomial probabilities) computed from `compute.binom.lik`
- `pi`, the probability of the genotypes $\pi_{1:K}$

The function should return a single $\log_e$ number. Note that in R, the default of the `log` function is the natural log (ln or $\log_e$).

```r
compute.loglik <- function(obs.lik, pi){

}
```

1.2.2. Compute the log likelihood for the input data Use the `compute.loglik` function from Section 1.2.1 to compute the log likelihood for the input data. To demonstrate the usage of this function, compute the log likelihood for the result from Section 1.1.2 assigned to `obs.lik` and the initial probabilities for genotypes set in Section 0.3, `pi.init` as input values.

Here is the call to the function and the output for you to check your code.

```r
loglik <- compute.loglik(obs.lik, pi.init)
loglik
```

```
## [1] -3993.978
```

2. Implement Functions for Genotype Inference and Parameter Estimation using EM

Total Points for Section: 20

2.1 Compute the responsibilities in the E-Step (Points: 5)

2.1.1 Write a function to compute the responsibilities

Write a function, called `compute.responsibilities`, that computes the probability of each locus i having every possible genotype $\forall k \in \{AA, AB, BB\}$,

$$\gamma(Z_i = k) = \frac{\pi_k Bin(x_i|N_i, \mu_k)}{\sum_{j=1}^{K} \pi_j Bin(x_i|N_i, \mu_j)}$$

Note: This probability is computed for each locus i and each genotype k.

This function should take as arguments:

- `obs.lik`, the likelihood (binomial probabilities) computed from `compute.binom.lik`
- `pi`, the probability of the genotypes $\pi_{1:K}$

The function should return a matrix of binomial probabilities with dimensions $T \times K$ where the rows reprsent the data points and the columns represent the genotypes. The denominator in the equation is the normalization constant, which leads to each row summing to 1.

```
compute.responsibilities <- function(obs.lik, pi){

}
```

2.1.2 Compute $\gamma(Z_{1:T})$ for the first EM iteration

Compute the responsibilities using the initial settings of the parameters `pi.init` and the binomial probabilities computed in Section 1.1.2, `obs.lik`. This is the basically the E-Step in the first iteration of EM.

Print out the log likelihood and the first 3 lines of the responsibility matrix ($\gamma(Z_{1:3})$).

Here is the code and output for you to check your code.

```
gamma <- compute.responsibilities(obs.lik, pi.init)
gamma[1:3, ]
```

```
##              [,1]         [,2]         [,3]
## [1,] 1.000000e+00 1.625561e-10 9.342696e-82
## [2,] 1.045455e-09 1.000000e+00 8.316505e-59
## [3,] 1.000000e+00 3.186425e-08 9.065209e-76
```

2.2 Updating the probability of genotypes (mixed weights) in the M-Step (Points: 5)

2.2.1 Write a function to update the parameter $\pi_{1:K}$

Write a function, called `update.pi`, that computes the update of π_k given the responsibilities $\gamma(Z_i)$ from the E-Step and the hyperparameter δ_k

$$\hat{\pi}_k = \frac{\left(\sum_{i=1}^{T} \gamma(Z_i = k)\right) + \delta(k) - 1}{\sum_{j=1}^{K} \left\{\left(\sum_{i=1}^{T} \gamma(Z_i = j)\right) + \delta(j) - 1\right\}}$$

This function should take as arguments:

- `gamma`, the responsibilities computed from `compute.responsibilities`

- `delta`, the hyperparameter for the Dirichlet prior, $\delta_{1:K}$

The function should return a vector $\hat{\pi}_{1:K}$ with K elements, one value for each genotype $k \in \{AA,\ AB,\ BB\}$.

```
update.pi <- function(gamma, delta){

}
```

2.2.2 Compute $\pi_{1:T}$ for the first EM iteration

Use `update.pi` to compute $\hat{\pi}$ for the first iteration of EM. Use the responsibilities computed in Section 2.1.2, `gamma` and hyperparameter for the prior of the initial state distribution, `deltaPi.hyper`. Print out the values of $\hat{\pi}_{1:K}$.

```
pi.hat <- update.pi(gamma, delta.hyper)
pi.hat
```

```
## [1] 0.793479840 0.202007532 0.004512628
```

2.3 Updating the binomial parameter $\mu_{1:K}$ in the M-Step (Points: 5)

2.3.1. Write a function to update the binomial parameter

Write a function, called `update.mu`, that computes the update of μ_k given the responsibilities $\gamma(Z_i)$ from the E-Step and the hyperparameters α_k and β_k

$$\hat{\mu}_k = \frac{\left(\sum_{i=1}^{T} \gamma(Z_i = k) x_i\right) + \alpha_k - 1}{\left(\sum_{i=1}^{T} \gamma(Z_i = k) N_i\right) + \alpha_k + \beta_k - 2}$$

This function should take as arguments:

- the responsibilities computed from `compute.responsibilities`
- the input reference base counts $x_{1:T}$
- the input depth $N_{1:T}$
- the hyperparameters for the Beta prior, $\alpha_{1:K}$ and $\beta_{1:K}$

The function should return a vector $\hat{\mu}_{1:K}$ with K elements, one value for each genotype $k \in \{AA,\ AB,\ BB\}$.

```
update.mu <- function(gamma, x, N, alpha, beta){

}
```

Hint: $\sum_{i=1}^{T} \gamma(Z_i = k) x_i$ can be computed using matrix multiplication. Since $\gamma(Z_{1:T})$ is a matrix with dimensions $T \times K$ and the transpose of $x_{1:T}$ (i.e. $(x_{1:T})^T$) is $1 \times T$, then $\boldsymbol{x^T} \times \boldsymbol{\gamma}$ will have dimensions $1 \times K$.

2.3.2 Compute the binomial parameter for the first EM iteration

Use `update.mu` to compute $\hat{\mu}_{1:K}$ for the first iteration of EM. Use the responsibilities computed in Section 2.1.2 and hyperparameters for the Beta prior, `alpha.hyper` and `beta.hyper`. Print out the values of $\hat{\mu}_{1:K}$.

```
mu.hat <- update.mu(gamma, x, N, alpha.hyper, beta.hyper)
mu.hat
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.9653329 0.6410865 0.1194937
```

2.4. Compute the log posterior (Points: 5)

2.4.1. Define a function, `compute.log.posterior`, that will compute the log posterior distribution

$$\log \mathbb{P} = \ell + \log Dirichlet(\hat{\boldsymbol{\pi}}|\boldsymbol{\delta}) + \sum_{k=1}^{K} \log Beta(\hat{\mu}_k|\alpha_k, \beta_k)$$

The function should take as argmuents:

- `loglik`, the log likelihood ℓ computed from `compute.loglik`
- `mu`, the binomial parameters $\mu_{1:K}$
- `pi`, the probability of the genotypes $\pi_{1:K}$
- `delta`, the hyperparameter for the Dirichlet prior, $\delta_{1:K}$,
- `beta`, the hyperparameters for the Beta prior, $\alpha_{1:K}$ and $\beta_{1:K}$

```
compute.log.posterior <- function(loglik, mu, pi, alpha, beta, delta){

}
```

Note that you can use the R function `dbeta` for the Beta distribution or `betapdflog` below. For the Dirichlet distribution, use the function below. `dirichletpdflog` accepts input vectors `pi` and `delta`, both of length `K`. The prior probabilities in the log posterior should be in log space.

```
dirichletpdflog <- function(pi, delta) {
    c <- lgamma(sum(delta, na.rm = TRUE)) - sum(lgamma(delta), na.rm = TRUE)  #normalizing constant
    l <- sum((delta - 1) * log(pi), na.rm = TRUE)  #likelihood
    return(c + l)
}

betapdflog <- function(x, alpha, beta){
  c <- -lbeta(alpha, beta)
  l <- (alpha - 1) * log(x) + (beta - 1) * log(1 - x)
  return(c + l)
}
```

2.4.2. Compute the log posterior for the input data Use the `compute.log.posterior` function to compute the log posterior for the input data. Use the updated parameters, `pi.hat` (Section 2.2.2) and `mu.hat` (Section 2.3.2). Use the hyperparameters, `delta.hyper`, `alpha.hyper`, and `beta.hyper` for the priors that you set in Section 0.3.

First you will need to recompute the `obs.lik` using `compute.binom.lik` and then the `loglik` using `compute.loglik` for the updated parameters, `mu.hat` and `pi.ha`.

Here are the calls to the functions and the output for you to check your code.

```
obs.lik <- compute.binom.lik(x, N, mu.hat)
loglik <- compute.loglik(obs.lik, pi.hat)
logP <- compute.log.posterior(loglik, mu.hat, pi.hat, alpha.hyper, beta.hyper, delta.hyper)
logP
```

```
## [1] -2725.843
```

The function `compute.log.posterior` should return a single value that will be used to monitor the EM algorithm at each iteration for convergence.

3. Run the Full EM algorithm

Total Points for Section: 25

3.1 Implement the full EM algorithm (Points: 10)

Implement the full EM algorithm for inferring the responsibilities and learning the binomial mixture model parameters in a Bayesian framework.

Refer to Lecture 2 Slide 12 for the EM algorithm. Run the EM algorithm until convergence, compute.log.posterior. Convergence is acheived when the log posterior changes by less than a user-defined valued, $\epsilon = 10^{-2}$.

i. Print out the converged parameters:

   - the converged parameters for the probability of the genotypes $\hat{\pi}_k$ for each genotype $k \in \{AA, AB, BB\}$.
   - the converged parameters for the probability of observing a reference base $\hat{\mu}_k$ for each genotype $k \in \{AA, AB, BB\}$.

ii. Save the the responsibilities $\gamma(Z_i)$ for the final EM iteration; you will need this for Section 3.2. Print out the responsibilities of the first 5 loci for the final EM iteration

iii. Store the log posterior for each iteration of EM; you will need this for Section 3.3. Print out the log posterior for all iterations.

Here are the converged parameters; use these value to check your code.

```
pi.hat
```

```
##         AA         AB         BB
## 0.75145895 0.19502661 0.05351445
```

```
mu.hat
```

```
##        AA        AB        BB
## 0.9705156 0.7817451 0.2640394
```

```
gamma[1:5, ]
```

```
##                AA          AB           BB
## [1,] 9.984052e-01 0.001594753 7.977912e-23
## [2,] 3.464552e-07 0.999999654 6.350446e-12
## [3,] 9.820998e-01 0.017900187 2.643911e-20
## [4,] 6.492569e-03 0.993507431 2.226454e-10
## [5,] 3.520319e-04 0.999647968 1.119168e-11
```

```
logP[1:iter]
```

```
## [1]      -Inf -2725.843 -2361.785 -2299.422 -2293.413 -2292.666 -2292.570
## [8] -2292.557 -2292.555
```

3.2 Determine the mutation status for the input data (Points: 5)

Using the responsibilities `gamma` from the final EM iteration, determine the mutation status (`NotSNV` or `SNV`) for each locus.

- Loci i is assigned `SNV` when $\gamma(Z_i = AB) + \gamma(Z_i = BB) >= 0.9$
- Loci i is assigned `NotSNV` when $\gamma(Z_i = AB) + \gamma(Z_i = BB) < 0.9$

Append this to the original table. Save the results to a file, named `Homework7_mutationCalls_FirstName-LastName.txt`, with the same columns as the input file (`counts`) and then appending an additional column with the mutation status (`NotSNV` or `SNV`)

```
* `chr`, `position`, `refBase`, `refCount`, `NRefBase`, `NrefCount`, `depth`
* `mutationStatus`
```

3.3. Plot the log posterior (Points: 5)

Plot the log posterior as a function of the EM iterations. The plot should show a monotonically increasing log posterior curve, which is a property of the EM algorithm. The x-axis should be the EM iteration number and y-axis is the log posterior.

3.4. Plot the binomial pmf for the converged parameters (Points: 5)

Plot the probability mass function (pmf) for each binomial distribution in the 3-component mixture using the converged parameters $\hat{\mu}_{1:K}$. Use $N = 40$ depth and a range of values $x = \{0, \ldots, 40\}$ reference read counts. The plot should look similar to the 3 colored binomial pmfs in Lecture 2, slide 24.