Miscellaneous Genomics Notes

- Post-imputation information measure[1]:
  - Let $G_{ij} \in \{0,1,2\}$ denote the genotype of the ith individual at the jth SNP in a study cohort of N samples. Let $p_{ijk} = P(G_{ij} = k|H,G)$ be the probability (obtained) from imputation that the genotype at the jth SNP of the ith individual is k. Define the expected allele dosage for the genotype at the jth SNP of the ith individual be
  $$e_{ij} = p_{ij1} + 2p_{ij2}$$
  Note that this equation may define $2e_{ij}$ elsewhere. Let $f_{ij} = p_{ij1} + 4p_{ij2}$, θ_j denote the unknown population allele frequency of the jth SNP with estimate
  $$\hat{\theta} = \frac{1}{2N}\sum_{i=1}^{N} e_{ij}$$
  and $X = \sum_{i=1}^{N} G_{ij}$.
    - The MACH $\hat{r}^2$ is the ratio of the empirically observed variance of the allele dosage to the expected binomial variance at Hardy-Weinberg equilibrium. At the jth SNP this is defined as
    $$\hat{r}_j^2 = \begin{cases} \dfrac{\frac{1}{N}\sum_{i=1}^{N} e_{ij}^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N} e_{ij}\right)^2}{2\hat{\theta}(1-\hat{\theta})} & \hat{\theta} \in (0,1) \\ 1 & \hat{\theta} \in \{0,1\} \end{cases}$$
    - The BEAGLE allelic R^2 is derived by approximating the R^2 between the best guess genotype (the most likely imputed genotype in the ith individual at the jth SNP, denoted by z_{ij}) and the allele dosage as an approximation of the true genotype in the case where the genotype is unknown. At the jth SNP this is defined as
    $$R_j^2 = \frac{\left[\sum_i z_{ij}e_{ij} - \frac{1}{N}\left(\sum_i z_{ij}\sum_i e_{ij}\right)\right]^2}{\left[\sum_i f_{ij} - \frac{1}{N}\left(\sum_i e_{ij}\right)^2\right]\left[\sum_i z_{ij}^2 - \frac{1}{N}\left(\sum_i z_{ij}\right)^2\right]}$$
    - The IMPUTE info measure is based on measuring the relative statistical information about the population allele frequency, θ_j, given by
    $$I_A = \begin{cases} 1 - \dfrac{\sum_{i=1}^{N}(f_{ij} - e_{ij}^2)}{2N(\hat{\theta}(1-\hat{\theta}))} & \hat{\theta} \in (0,1) \\ 1 & \hat{\theta} \in \{0,1\} \end{cases}$$
  - The SNPTEST INFO measure[2] is similar to the IMPUTE info measure when assuming an additive model (but not dominant model). It reflects the information in imputed genotypes relative to the information if only the allele frequency were known, defined as:
  $$\text{INFO} = 1 - \frac{1}{N}\sum_{i=1}^{N}\frac{\sqrt{\hat{\theta}(1-\hat{\theta})}}{\sqrt{\theta(1-\theta)}}$$

[1] http://www.nature.com/articles/nrg2796.

[2] https://www.chg.ox.ac.uk/~gav/snptest/#info_measures.

where the numerator $\hat{\theta}$ is the imputed allele frequency and the denominator employs the estimated allele frequency of a given SNP across individuals (assuming HWE):

$$\theta = \frac{\sum_i \mathrm{DS_i}}{2\sum_{i,g} P(g_i = g)} \qquad g \in \{0,1\}$$

Note that INFO score equals 1 exhibits perfect imputation, and it could drop below 0.

– The MACH, BEAGLE and IMPUTE measures seem to be highly correlated with BEAGLE R^2 systemically reporting lower values and undefined at 3% of the SNPs and MACH r^2 often exceeds 1.

Algorithms and Technicals

- The Metropolis-Hastings Algorithm:
  - Suppose we want to sample a target distribution where we don't know its normalising constant, which we denote as $p(\theta)$, but we have a known distribution $g(\theta)$ such that it satisfies $p(\theta) \propto g(\theta)$. The Metropolis-Hastings algorithm is as follows:
    1. Select initial value θ_0.
    2. For $i = 1, \cdots, m$, repeat:
       (a) Draw candidate $\theta^* \sim q(\theta^* \mid \theta_{i-1})$.
       (b) Calculate α with
       $$\alpha = \frac{g(\theta^*)/q(\theta^* \mid \theta_{i-1})}{g(\theta_{i-1})/q(\theta_{i-1} \mid \theta^*)} = \frac{g(\theta^*)q(\theta_{i-1} \mid \theta^*)}{g(\theta_{i-1})q(\theta^* \mid \theta_{i-1})}$$
       (c) Determine whether to retain the sample θ^* based on α:
         * If $\alpha \geq 1$, accept θ^* and set $\theta_i \leftarrow \theta^*$;
         * If $0 < \alpha < 1$:
           · Accept θ^* and set $\theta_i \leftarrow \theta^*$ with probability α;
           · Reject θ^* and set $\theta_i \leftarrow \theta_{i-1}$ with probability $1 - \alpha$.
  - This algorithm is a valid Markov chain. One can pick q such that:
    * q does not depend on the previous draw θ_{i-1}, in this case we need to have q is similar to p.
    * q depends on the previous draw, where we have a random walk Metropolis-Hastings.
  - If we q is normal, we can increase its standard deviation for decreasing acceptance rate. Targeted acceptance rate can be $23\%-50\%$.

- Gibbs sampling is to sample from an unknown distribution if multiple parameters are unknown. When we sample one of the parameters, simply treat the other as a known constant (due to the chain rule of conditional probability).
  - Let's assume the unknown distribution is $p(\theta, \varphi \mid y)$ but we have $g(\theta, \varphi)$ that satisfies $p(\theta, \varphi \mid y) \propto g(\theta, \varphi)$. The Gibbs sampler is as follows:
    1. Select initial value θ_0 and φ_0.
    2. For $i = 1, \cdots, m$, repeat the following as one Gibbs cycle:
       (a) Using φ_{i-1} to draw $\theta_i \sim p(\theta \mid \varphi_{i-1}, y)$.
       (b) Using θ_i to draw $\varphi_i \sim p(\varphi \mid \theta_i, y)$.
       (c) Update (θ_i, φ_i) according to the Metropolis-Hastings algorithm.

- Autocorrelation:
  - Autocorrelation is an important structure to inspect in order to determine the validity of a MCMC simulation. If autocorrelation is high for large number of lags (one can inspect the autocorrelation plot), it probably indicates that the simulation hasn't converged to a stationary distribution, and we need to increase the number of samples drawn from the chain or modify model hyperparameters.

- Autocorrelation is also important in calculating the effective sample size of our MCMC. The effective sample size is how many independent samples from the stationary distribution you would have to draw to have equivalent information in our MCMC. It is essentially the sample size we choose for our Monte Carlo estimation.
- Broaderly speaking, an effective sample size is often concerned in two scenarios, that is, when our data is autocorrelated or weighted. An intuitive example would be if our $X_1 = \cdots = X_n$ and thus perfectly autocorrelated, we basically just have one sample; or if our samples are weighted such that $w_1 = 1$ and $w_2 = \cdots = w_n = 0$, the effective sample size is 1 as well. Mathematically, it is defined as the number of i.i.d. samples required to achieve the same level of variance.
- If one only cares the mean of the posterior, an effective sample size of the scale 100 to 1000 is good enough; if one seeks to create a confidence interval, then several thousands of effective samples are required.

• Hidden Markov Model:

- Let us denote the observation sequence up to time T as $O = \{O_1, O_2, \cdots, O_T\}$. An HMM consists of 5 components N, M, A, B, and π:
  * N is the number of all possible "hidden" states at any time t.
  * M is the number of all observable states emitted from the hidden states at any time t.
  * A is the transition matrix for the hidden states.
  * B is the emission matrix for observing the observables from the hidden states, varied according to states.
  * π is the initial probability distribution of the hidden states at the starting point.
- Three key HMM problems are posed here:
  1. What is the probability that a model generated a sequence of observations $O = \{O_1, O_2, \cdots, O_T\}$ given the input parameters $\lambda = (A, B, \pi)$? Namely we want to calculate $P(O \mid \lambda)$.

  ⇒ A naive enumerating way of calculating the probability could be

$$\begin{aligned} P(O \mid \lambda) &= \sum_Q P(O \mid Q, \lambda) P(Q \mid \lambda) \\ &\sim \mathscr{O}(TN^T) \end{aligned}$$

  But this is computationally prohibitive. Instead, we could use the forward function $\alpha_t(i)$ where

$$\alpha_t(i) = P(O_1, \cdots, O_t, q_t = S_i \mid \lambda)$$

  With dynamic programming, we could solve this inductively:

$$\begin{aligned} \alpha_1(i) &= \pi_i b_i(O_1) \\ \alpha_{t+1}(j) &= \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \\ P(O \mid \lambda) &= \sum_{i=1}^N \alpha_T(i) \\ &\sim \mathscr{O}(N^2 T) \end{aligned}$$

We also introduce the backward function $\beta_t(i)$ where

$$\beta_t(i) = P(O_{t+1}, \cdots, O_T \mid q_t = S_i, \lambda)$$

with the following induction:

$$\begin{aligned} \beta_T(i) &= 1 \\ \beta_t(i) &= \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \end{aligned}$$

2. What sequence $Q = \{q_1, q_2, \cdots, q_T\}$ of states best explains a sequence of observations $O = \{O_1, O_2, \cdots, O_T\}$?

⇒ Denote $\gamma_t(i)$ as the probability of being in state S_i at time t, we can easily calculate this as

$$\begin{aligned} \gamma_t(i) &= P(q_t = S_i \mid O, \lambda) \\ &= \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \end{aligned}$$

An intuitive thought about γ would be dividing the probability of getting the full sequence of observation up to time T given the current state over the total probability of observing that sequence. It is easy to observe that $\sum_{i=1}^{N} \gamma_t(i) = 1$.

Now, it becomes natural to consider at each $t \leq T$, what is the maximum likely state the observable is in. However, considering this sequential maximum does not necessarily make sense, at they may not form a valid path (path with non-zero probability). Hence, the better way of considering this is to find the most likely state sequence, namely $P(Q \mid O, \lambda)$.

The resulting algorithm that solves this problem is called the Viterbi algorithm. Denote

$$\delta_t(i) = \max_{q_1, \cdots, q_{t-1}} P(\{q_1, \cdots, q_{t-1}, q_t = i\}, \{O_1, \cdots, O_t\} \mid \lambda)$$

as the path with the highest probability that accounts for the first t observations and ends at state S_i. We use $\psi_t(i)$ to track the state at t that maximises $\delta_t(i)$. The Viterbi algorithm can then be formulated as

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(O_1) & \psi_1(i) &= 0 \\ \delta_t(j) &= \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \cdot b_j(O_t) & \psi_t(j) &= \arg\max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \\ P^* &= \max_{1 \leq i \leq N} [\delta_T(i)] & q_T^* &= \arg\max_{1 \leq i \leq N} [\delta_T(i)] \end{aligned}$$

And we have $q_t^* = \psi_{t+1}(q_{t+1}^*)$ where here the $*$ is to denote the optimal variables.

3. Given a set of observation sequences $O = \{O_1, O_2, \cdots, O_T\}$, how do we learn the model parameters $\lambda = (A, B, \pi)$ that would generate them?

⇒ This can be done with the Baum-Welch Algorithm. It is an expectation-maximisation (EM) method (a.k.a., gradient "descent"), iterative, and converges to a local optimum.

Denote $\xi_t(i,j)$ as

$$\begin{aligned} \xi_t(i,j) &= P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \end{aligned}$$

It is notable that $\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$. Now, to estimate the parameter, we could use

$$\begin{aligned}
\hat{\pi} &= \gamma_1(i) \\
\hat{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\
\hat{b}_j(k) &= \frac{\sum_{t=1,\text{ s.t. } O_t = v_k}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}
\end{aligned}$$

We are then able to iteratively update our model parameters λ with the previous observations, and update the observations with the new parameters, until we reach a local maximum or minimum (which is the essence of EM).