

Mixture MCMC Algorithm (SEMPPR ROC)

May 13, 2015

1 Draw conditional ϕ values, p values, and z values

```
double logLikelihood = 0.0;
int numGenes = genome.getGenomeSize();
unsigned numMixtures = parameter.getNumMixtureElements();
double dirichletParameters[numMixtures];

for(int i = 0; i < numGenes; i++)
{
    Gene gene = genome.getGene(i);
    double currLike = 0.0;
    double propLike = 0.0;

    double probabilities[numMixtures];

    for(unsigned k = 0; k < numMixtures; k++)
    {
        double* logLikRatio =
            model.calculateLogLikelihoodRatioPerGene(gene, i,
                parameter, k);
        //double check probabillites

        // p_i * f(X | Theta, Phi)
        probabilities[k] =
            parameter.getCategoryProbability(k, iteration/thining) *
            std::exp(logLikRatio_current);
        currLike += probabilities[k];
        propLike += parameter.getCategoryProbability(k, i) *
            std::exp(logLikRatio_proposed);
    }

    // Get category in which the gene is placed in.
    // If we use multiple sequence observrvation
    // randMultinom needs an parameter N to place N
    // observations in numMixture buckets
}
```

```

unsigned categoryOfGene =
    ROCParameter::randMultinom(probabilities , numMixtures);
parameter.setMixtureAssignment(i , categoryOfGene);
dirichletParameters[categoryOfGene] += 1;

// accept/reject proposed phi values
double r = unif(0,1);
if( r < (propLike / currLike) )
{
    // moves proposed phi to current phi
    parameter.updateExpression(i);
    logLikelihood += propLike;
else{
    logLikelihood += currLike;
    }
if((iteration % thinning) == 0)
{
    parameter.updateExpressionTrace(iteration/thinning , i);
}
}

// get new mixture probabilities (gamma = 0)
double newMixtureProbabilities[numMixtures];
ROCParameter::randDirichlet(dirichletParameters ,
    numMixtures, newMixtureProbabilities);
for(unsigned k = 0u; k < numMixtures; k++)
{
    parameter.setCategoryProbability(k, iteration/thinning ,
        newMixtureProbabilities[k]);
}
return logLikelihood;

```

2 Draw conditional codon specific parameter values

```

unsigned numMixtures = parameter.getNumMixtureElements();

logAcceptanceRatioPerMixture[22];

// for all amino acids (incl. STOP and Ser2)
// but skip one codon AA and STOP
for(unsigned aa = 0; aa < 22; i++)
{
    for(int i = 0; i < numGenes; i++)
    {
        // which mixture element does the gene belong to
        unsigned mixtureElement = parameter.getMixtureOfGene(i);
    }
}

```

```

// how is the mixture element defined
unsigned mutationCategory =
    parameter.getMutationCategory(mixtureElement);
unsigned selectionCategory =
    parameter.getSelectionCategory(mixtureElement);
unsigned expressionCategory =
    parameter.getExpressionCategory(mixtureElement);

// get parameter values for mixture element
double phiValue =
    parameter.getExpression(geneIndex, expressionCategory, false);
double mutation[numCodons - 1];
parameter.getParameterForCategory(mutationCategory,
    ROCParameter::dM, curAA, false, mutation);
double selection[numCodons - 1];
parameter.getParameterForCategory(selectionCategory,
    ROCParameter::dEta, curAA, false, selection);

// get proposed values for mixture element
double mutation_proposed[numCodons - 1];
parameter.getParameterForCategory(mutationCategory,
    ROCParameter::dM, curAA, true, mutation_proposed);
double selection_proposed[numCodons - 1];
parameter.getParameterForCategory(selectionCategory,
    ROCParameter::dEta, curAA, true, selection_proposed);

int codonCount[numCodons];
obtainCodonCount(seqsum, curAA, codonCount);

logLikelihood += calculateLogLikelihoodPerAAPerGene(numCodons,
    codonCount, seqsum, mutation, selection, phiValue);
logLikelihood_proposed +=
    calculateLogLikelihoodPerAAPerGene(numCodons,
    codonCount, seqsum, mutation_proposed, selection_proposed,
    phiValue);
}
logAcceptanceRatioPerMixture[aa] = logLikelihood_proposed -
    logLikelihood;
}

```

3 Draw conditional codon specific parameter values

```

double logProbabilityRatio = 0.0;

double currentSphi = parameter.getSphi(false);

```

```

double currentMPhi = -(currentSphi * currentSphi) / 2;

double proposedSphi = parameter.getSphi(true);
double proposedMPhi = -(proposedSphi * proposedSphi) / 2;

for(int i = 0; i < numGenes; i++)
{
    // make sure to use the phi value
    // for the mixture element the gene
    // is currently assigned to
    unsigned mixture = parameter.getMixtureAssignment(i);
    double phi = parameter.getExpression(i, mixture, false);
    logProbabilityRatio +=
        std::log(ROCPParameter::densityLogNorm(phi,
            proposedMPhi, proposedSphi)) -
        std::log(ROCPParameter::densityLogNorm(phi,
            currentMPhi, currentSphi));
}

logProbabilityRatio -= (std::log(currentSphi) -
    std::log(proposedSphi));

if( -ROCPParameter::randExp(1) < logProbabilityRatio )
{
    // moves proposed Sphi to current Sphi
    parameter.updateSphi();
}
if((iteration % thinning) == 0)
{
    parameter.updateSphiTrace(iteration/thinning);
}

```