STAT 535A: Monte Carlo Methods
Assignment on Blang

Due on Wednesday, April 4, 2018

Creagh Briercliffe

creagh@stat.ubc.ca

Acknowledgement: this work was done in collaboration with Saif, Vaden and Esten.

Implementing samplers over matchings

Basic sampler for permutations

The method `sampleUniform()`, in the file `Permutation.xtend`, implements a sampler for permutations. This method takes a given configuration, sorts this configuration, then shuffles the elements. It iterates through the configuration, choosing swaps with other elements, uniformly at random. The result is an independent permutation, sampled uniformly at random, and constructed in-place.

A more basic sampler for permutations might choose two elements uniformly at random and swap them. To establish irreducibility and invariance with respect to the uniform distribution on the space of permutations, consider the following. Let π be the uniform distribution over the space of permutations of size n, denoted $\mathbb{X}_n$. Hence, for any $x \in \mathbb{X}_n$, $\pi_x = 1/|\mathbb{X}_n|$. For $x, y \in \mathbb{X}_n$, we define the Markov kernel for this sampler as $K_{xy} = n^{-2}$, if x and y differ by having only two elements swapped, and 0 otherwise. Consequently, we have that $K_{xy} = K_{yx}$. Therefore, we see that K is π-reversible, since $\forall x, y \in \mathbb{X}_n$, $\pi_x K_{xy} = \pi_y K_{yx}$. This also implies that K is π-invariant. Lastly, K is irreducible since $\forall x, y \in \mathbb{X}_n$, state y can be reached from state x through a finite number of swaps, for $n < \infty$.

Non-uniform case

I construct a sampler for non-uniformly distributed permutations using a simple Metropolis-Hastings algorithm with uniform proposal. See `PermutationSampler.java` for the implementation.

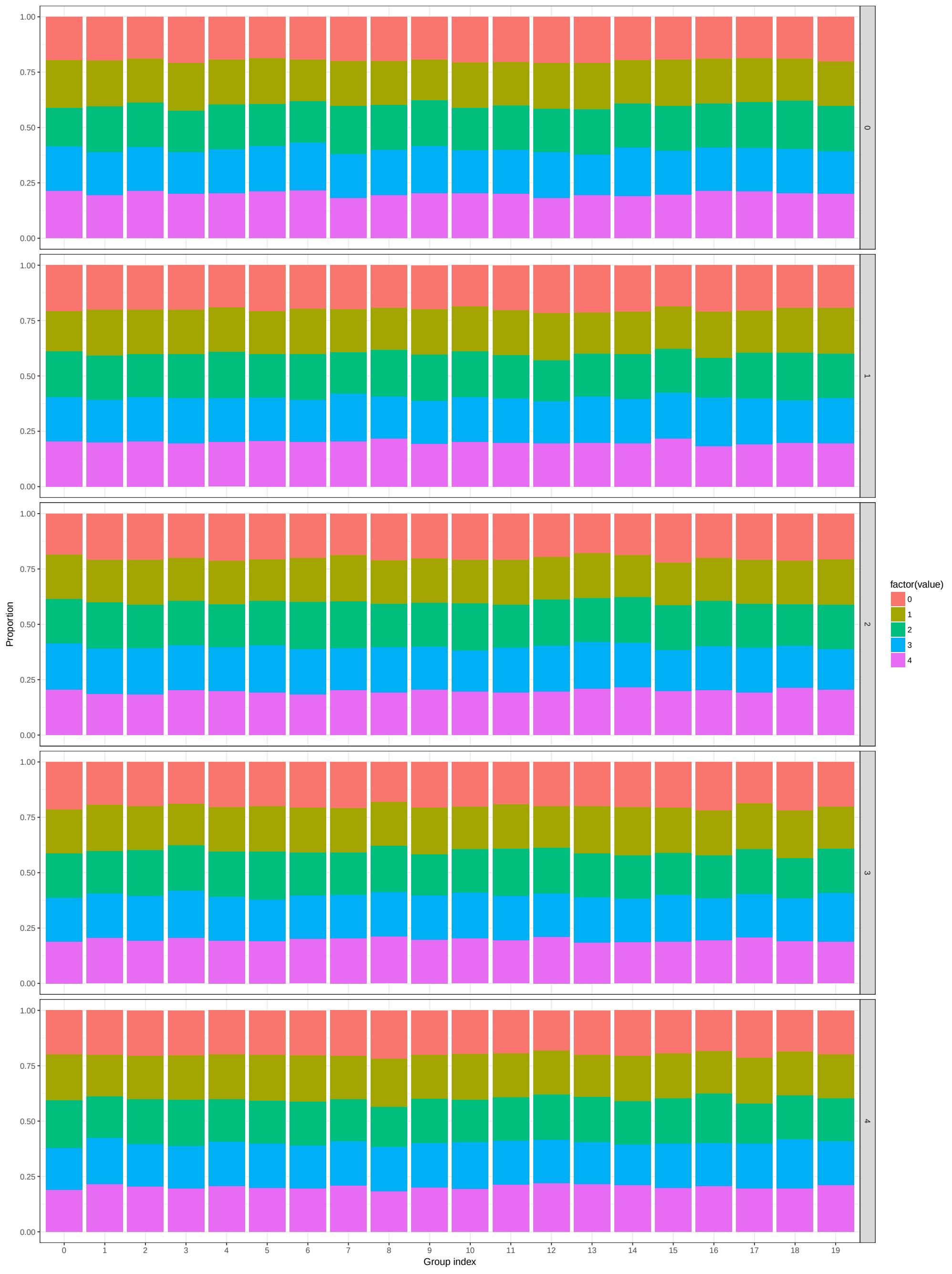
Understanding the test

The file `ExhaustiveDebugRandom.class` implements a superclass, extending `Random.class`. It is used for fully discrete probability models, allowing the user to compute the transition probabilities of a sampling process. An object of this class gets instantiated in files like `DiscreteMCTest.java`. At a high-level, this class can be used to construct the decision tree induced by a discrete random process.

The file `DiscreteMCTest.java` tests the irreducibility and invariance of a discrete Markov Chain. At a high-level, the code executes as follows:

- Enumerate the states in the state space.
- Compute the transition probabilities of the Markov Chain, using an object of the `ExhaustiveDebugRandom.class`.
- Compute the target distribution.
- Use the transition matrix and target distribution to check invariance. Each kernel is tested separately to pinpoint errors.
- Check irreducibility: mix all kernels together and determine if all states are reachable from some arbitrarily chosen starting state.

A statistical model involving a combinatorial space


Proportion
Group index
factor(value)