Algorithm 1 Gibbs Sampling

1 Gibbs Sampling in R

Listing 1: Gibbs Sampling Function in R

```r
gibbs_sampling <- function(dataset, iterations, csi, g) {
  # Get the size of the binary vector J
  data_size <- ncol(dataset)

  # Initialize a binary vector J_init
  J_init <- sample(0:1, size = data_size, replace = TRUE)

  # List to store the sample of binary vectors
  sample_list <- list()

  # Perform Gibbs sampling for the specified number of
      iterations
  for (iteration in 1:iterations) {
    # Iterate over each element in the binary vector J
    for (i in 1:(data_size - 1)) {
      # Append the current state of the binary vector to
          the sample
      sample_list[[length(sample_list) + 1]] <- J_init

      # Toggle the i-th element and calculate the
          probability
      J_init[i] <- 1
      probability <- exp(csi * g(J_init))
      J_init[i] <- 0
      probability <- probability / (exp(csi * g(J_init)) +
          probability)

      # Generate a random sample from a Bernoulli
          distribution with the calculated probability
      J_init[i] <- rbinom(1, 1, probability)
    }
  }

  # Return the sample of binary vectors
  return(sample_list)
}

# Example usage:
# Replace dataset, iterations, csi, and g with your actual
    data and functions
# result <- gibbs_sampling(dataset, iterations, csi, g)
```

function GibbsSampling(data, itts, ξ, g)
    $data\_size \leftarrow$ length of vectors
    $J \leftarrow$ random initialisation
    $sample \leftarrow$ new Array
    for itt $\leftarrow 1$ to itts **do**
        for $i \leftarrow 1$ to $data\_size - 1$ **do**
            J gets added to the sample
            $J[i] \leftarrow 1$
            $p_C(J_s = 1, J_{-s}) \leftarrow \exp(\xi \cdot g(J))$
            $J[i] \leftarrow 0$
            $p_C(J_s = 1 \mid J_{-s}) \leftarrow \frac{p_C(J_s=1,J_{-s})}{\underbrace{\exp(\xi \cdot g(J))}_{1} + p_C(J_s=1,J_{-s})}$