library(nimble)

# Define the model code with alpha as a function of covariates

code <- nimbleCode({

for (i in 1:N) {

for (k in 1:K) {

# Linear model for alpha, with covariate X influencing alpha

log(alpha[i, k]) <- beta0[k] + beta1[k] \* X[i]

}

# Dirichlet distribution with alphas depending on covariates

probs[i, 1:K] ~ ddirch(alpha[i, 1:K])

}

# Priors for beta coefficients

for (k in 1:K) {

beta0[k] ~ dnorm(0, sd = 10) # Prior for intercept

beta1[k] ~ dnorm(0, sd = 10) # Prior for slope

}

})

# Set number of categories and samples

K <- 3 # Number of classes

N <- 10 # Number of samples (observations)

# Create a covariate vector (e.g., some covariate for each sample)

X <- rnorm(N, mean = 0, sd = 1) # Example covariate values

# Initialize the data

probs <- matrix(NA, nrow = N, ncol = K) # Placeholder for the probabilities

# Constants and data for the model

constants <- list(N = N, K = K)

data <- list(probs = probs, X = X)

# Initial values for parameters

inits <- list(beta0 = rep(0, K), beta1 = rep(0, K))

# Build the model

dirichlet\_cov\_model <- nimbleModel(code = code, constants = constants, data = data, inits = inits)

# Compile the model

compiled\_model <- compileNimble(dirichlet\_cov\_model)

# Configure and run MCMC (if needed)

mcmcConf <- configureMCMC(dirichlet\_cov\_model)

mcmc <- buildMCMC(mcmcConf)

compiled\_mcmc <- compileNimble(mcmc, project = dirichlet\_cov\_model)

samples <- runMCMC(compiled\_mcmc, niter = 10000)

**Explanation:**

* **Covariate Influence**:
  + In the loop for k, each α[i,k] is modeled as a log-linear function of a covariate X[i]. The log-link function (log(alpha[i, k])) ensures that α[i,k] remains positive.
  + You can modify the linear model to include more covariates or interaction terms as needed.
* **Priors**:
  + The priors on beta0[k] (intercept) and beta1[k] (slope for covariate X) are assumed to be normal, but you can adjust them based on your problem.
* **Dirichlet**:
  + The probabilities probs[i, 1:K] are drawn from a Dirichlet distribution where the α's depend on covariates.

**Running Inference:**

After building the model, you can run MCMC or other inference methods on the model just like in the simpler case. This approach allows you to estimate how the covariates influence the concentration parameters of the Dirichlet distribution, and thus the probability vectors.