

```

data {
  int<lower = 0> Np;           // 処理区の数
  int<lower = 0> Ns;           // 小区画の数
  int<lower = 0> Nsc;         // 小区画がある区画（地がき区）の数
  int<lower = 0> Nsp[Nsc];    // 地がき区の小区画数
  int<lower = 0> Nq;           // 方形区の数
  int<lower = 0> Nscq;        // 小区画がある方形区の数
  int<lower = 0> Nspc;        // 稚樹の種数
  vector<lower = 0>[Nq] Height; // 下層植生の平均群落高 (m)
  int<lower = 0, upper = 1> Y[Nq, Nspc]; // 平均群落高以上の稚樹のありなし
  int<lower = 0, upper = 1> Treat1[Nq]; // 地がき区: t=1
  int<lower = 0, upper = 1> Treat2[Nq]; // 根返し区: t=2
  int<lower = 0, upper = 1> Treat3[Nq]; // 伐根区: t=3
  int<lower = 0, upper = 1> Treat4[Nq]; // 林床区: t=4
  int<lower = 0, upper = 1> Fence[Nq]; // シカ柵: x(0:なし, 1:あり)
  int<lower = 1, upper = Np> Plot[Nq]; // 区画インデックス
  int<lower = 1, upper = Ns> Subplot[Nscq]; // 小区画インデックス
}

```

```

parameters {
  vector[4] beta_0;           // 稚樹有無の処理区ごとの切片(logit):  $\beta_0$ 
  real beta_0H;               // 稚樹有無への下層植生の効果:  $\beta_{0H}$ 
  real beta_0F;               // 稚樹有無へのシカ柵の効果:  $\beta_{0F}$ 
  vector[Np] epsilon_0P;      // 稚樹有無の処理区の変量効果:  $\epsilon_{0P}$ 
  vector[Ns] epsilon_0Q;      // 稚樹有無の小区画の変量効果:  $\epsilon_{0Q}$ 
  matrix[4, Nspc] epsilon_0S; // 稚樹有無の樹種の変量効果:  $\epsilon_{0S}$ 
  vector<lower = 0>[4] beta_H; // 下層植生平均群落高の処理区ごとの切片:  $\beta_H$ 
  real beta_HF;               // 平均群落高へのシカ柵の効果:  $\beta_{HF}$ 
  vector[Np] epsilon_HP;      // 平均群落高の処理区の変量効果:  $\epsilon_{HP}$ 
  vector[Ns] epsilon_HQ;      // 平均群落高の小区画の変量効果:  $\epsilon_{HQ}$ 
  vector<lower = 0>[9] sigma; // 標準偏差:  $\sigma, \sigma_{HP}, \sigma_{HQ}, \sigma_{0P}, \sigma_{0Q}, \sigma_{0S}$ 
}

```

```

transformed parameters {
  vector[Nq] mu;
  matrix[Nq, Nspc] logit_omega;

  for (n in 1:Nscq) { // 地がき区
    mu[n] = beta_H[1] * Treat1[n]
      + beta_H[2] * Treat2[n]
      + beta_H[3] * Treat3[n]

```

```

    + beta_H[4] * Treat4[n]
    + beta_HF * Fence[n]
    + epsilon_HP[Plot[n]]
    + epsilon_HQ[Subplot[n]];
for (s in 1:Nspc) {
  logit_omega[n, s] = (beta_0[1] + epsilon_OS[1, s]) * Treat1[n]
    + (beta_0[2] + epsilon_OS[2, s]) * Treat2[n]
    + (beta_0[3] + epsilon_OS[3, s]) * Treat3[n]
    + (beta_0[4] + epsilon_OS[4, s]) * Treat4[n]
    + beta_OH * mu[n]
    + beta_OF * Fence[n]
    + epsilon_OP[Plot[n]]
    + epsilon_OQ[Subplot[n]];
}
}
for (n in (Nscq + 1):Nq) { // 地がき区以外
  mu[n] = beta_H[1] * Treat1[n]
    + beta_H[2] * Treat2[n]
    + beta_H[3] * Treat3[n]
    + beta_H[4] * Treat4[n]
    + beta_HF * Fence[n]
    + epsilon_HP[Plot[n]];
  for (s in 1:Nspc) {
    logit_omega[n, s] = (beta_0[1] + epsilon_OS[1, s]) * Treat1[n]
      + (beta_0[2] + epsilon_OS[2, s]) * Treat2[n]
      + (beta_0[3] + epsilon_OS[3, s]) * Treat3[n]
      + (beta_0[4] + epsilon_OS[4, s]) * Treat4[n]
      + beta_OH * mu[n]
      + beta_OF * Fence[n]
      + epsilon_OP[Plot[n]];
  }
}
}

model {
  for (n in 1:Nq) {
    Height[n] ~ normal(mu[n], sigma[1]) T[0, ];
    for (s in 1:Nspc) {
      Y[n, s] ~ bernoulli_logit(logit_omega[n, s]);
    }
  }
}

```

```

beta_H ~ normal(0, 10);
beta_HF ~ normal(0, 10);
epsilon_HP ~ normal(0, sigma[2]);
epsilon_HQ ~ normal(0, sigma[3]);
beta_0 ~ normal(0, 10);
beta_0H ~ normal(0, 10);
beta_0F ~ normal(0, 10);
epsilon_OP ~ normal(0, sigma[4]);
epsilon_OQ ~ normal(0, sigma[5]);
for (i in 1:4) {
  epsilon_OS[i] ~ normal(0, sigma[i + 5]);
}
sigma ~ normal(0, 5);
}

```

```

generated quantities {
  vector[4] gamma_H;
  vector[4] gamma_0;
  real gamma_OHF = beta_0H * beta_HF;
  real gamma_0F = beta_0H * beta_HF + beta_0F;

  gamma_H[1] = beta_H[1] - beta_H[4]; // 林床vs地がき
  gamma_H[2] = beta_H[2] - beta_H[4]; // 林床vs根返し
  gamma_H[3] = beta_H[1] - beta_H[3]; // 伐根vs地がき
  gamma_H[4] = beta_H[2] - beta_H[3]; // 伐根vs根返し

  gamma_0[1] = beta_0[1] - beta_0[4]; // 林床vs地がき
  gamma_0[2] = beta_0[2] - beta_0[4]; // 林床vs根返し
  gamma_0[3] = beta_0[1] - beta_0[3]; // 伐根vs地がき
  gamma_0[4] = beta_0[2] - beta_0[3]; // 伐根vs根返し
}

```