Stan code of gamma survival model

```
data{
  int N; // number of nests
  int decayed[N]; // id for censored nests "0", and fully decayed nests "1"
  int days_to_event[N]; // days until full decay (or until last observation day if
censored)
  int HB[N]; // habitat type
  int ES[N]; // nest exposure
  int P [N]; // construction type
  int R [N]; // average rain
  int HT[N]; // nest height
  int M [N]; // integrated nest (M stands for mixed)
  int DR[N]; // rain at dusk
  int MT[N]; // min temperature
  int S [N]; // average storms
  int SP[N]; // tree species
  }

parameters{
  vector[2]  hb; // intercept varying by habitat type
  vector[2]  es; // intercept varying by exposure
  vector[2]  p;  // intercept varying by position
  vector[2]  m;  // intercept varying by construction type
  vector[3]  ht; // intercept varying by height
  vector[3]  r;  // intercept varying by average precipitation
  vector[3]  mt; // intercept varying by minimum temperature
  vector[2]  dr; // intercept varying by rain at dusk
  vector[3]  s;  // intercept varying by average number of stprms
  vector[11] sp; // intercept varying species
  real<lower=0> phi; // variance for computing "shape" and "rate", from mean "mu""
}

transformed parameters{
vector<lower=0>[N] mu; // average decomposition

for (i in 1:N) {

    mu[i] =
hb[HB[i]]+es[ES[i]]+p[P[i]]+m[M[i]]+ht[HT[i]]+r[R[i]]+mt[MT[i]]+dr[DR[i]]+s[S[i]]+sp[SP
[i]]; // linear model for average
    mu[i] = exp(mu[i]; //link function : exp of mu => positive

  }
}

model{
////priors
  hb ~ normal(0,5);
  es ~ normal(0,5);
  p  ~ normal(0,5);
  m  ~ normal(0,5);
  ht ~ normal(0,5);
  r  ~ normal(0,5);
  mt ~ normal(0,5);
  dr ~ normal(0,5);
  s  ~ normal(0,5);
  sp ~ normal(0,5);
  phi~ gamma (0.1,0.1);

//we model observed nests using gamma distribution

 for ( i in 1:N )
   if(decayed[i]==1) days_to_event[i]~gamma( mu[i] * phi, phi);
```

```
// and we model censored nests, using complementary cumulative gamma distribution

 for ( i in 1:N)
 if(decayed[i]==0) target+=gamma_lccdf(days_to_event[i] | mu[i] * phi, phi);

}

generated quantities{

real Decay; // Average decomposition time
vector [N] log_lik; // log likelihood for model diagnostic and comparison

//// MAIN EFFECTS
matrix [N,2] D_HB;
matrix [N,2] D_ES;
matrix [N,2] D_P;
matrix [N,2] D_M;
matrix [N,3] D_HT;
matrix [N,3] D_R;
matrix [N,3] D_S;
matrix [N,3] D_MT;
matrix [N,2] D_DR;
matrix[N,11] D_SP;

//// AVERAGE MAIN EFFECTS
vector [2]   Dec_HB;
vector [2]   Dec_ES;
vector [2]   Dec_P;
vector [2]   Dec_M;
vector [3]   Dec_HT;
vector [3]   Dec_R;
vector [3]   Dec_S;
vector [3]   Dec_MT;
vector [2]   Dec_DR;
vector[11]   Dec_SP;

//// Calculations

Decay = mean(mu);

for (i in 1:N){
  for(j in 1:2){
    D_HB[i,j]  = exp(es[ES[i]] + p[P[i]] + m[M[i]] + dr[DR[i]] + ht[HT[i]] + r[R[i]] +
s[S[i]] + mt[MT[i]] + sp[SP[i]] +  hb[j]);
    D_ES[i,j]  = exp(hb[HB[i]] + p[P[i]] + m[M[i]] + dr[DR[i]] + ht[HT[i]] + r[R[i]] +
s[S[i]] + mt[MT[i]] + sp[SP[i]] +  es[j]);
    D_P [i,j]  = exp(hb[HB[i]] + es[ES[i]] + m[M[i]] + dr[DR[i]] + ht[HT[i]] + r[R[i]]
+ s[S[i]] + mt[MT[i]] + sp[SP[i]] + p[j]);
    D_M [i,j]  = exp(hb[HB[i]] + es[ES[i]] + p[M[i]] + dr[DR[i]] + ht[HT[i]] + r[R[i]]
+ s[S[i]] + mt[MT[i]] + sp[SP[i]] + m[j]);
    D_DR[i,j]  = exp(hb[HB[i]] + es[ES[i]] + m[M[i]] + p[P[i]] + ht[HT[i]] + r[R[i]] +
s[S[i]] + mt[MT[i]] + sp[SP[i]] +  dr[j]);

  }

  for(j in 1:3){
    D_HT[i,j]  = exp(es[ES[i]] + p[P[i]] + m[M[i]] + dr[DR[i]] + hb[HB[i]] + r[R[i]] +
s[S[i]] + mt[MT[i]] + sp[SP[i]]  + ht[j]);
    D_R [i,j]  = exp(es[ES[i]] + p[P[i]] + m[M[i]] + dr[DR[i]] + hb[HB[i]] + ht[HT[i]]
+ s[S[i]] + mt[MT[i]] + sp[SP[i]] + r[j]);
    D_S [i,j]  = exp(es[ES[i]] + p[P[i]] + m[M[i]] + dr[DR[i]] + hb[HB[i]] + r[R[i]] +
ht[HT[i]] + mt[MT[i]] + sp[SP[i]] + s[j]);
    D_MT[i,j]  = exp(es[ES[i]] + p[P[i]] + m[M[i]] + dr[DR[i]] + hb[HB[i]] + r[R[i]] +
s[S[i]] + ht[HT[i]] + sp[SP[i]]  + mt[j]);
  }

  for(j in 1:11){
    D_SP[i,j]  = exp(es[ES[i]] + p[P[i]] + m[M[i]] + dr[DR[i]] + ht[HT[i]] + r[R[i]] +
s[S[i]] + mt[MT[i]] + hb[HB[i]] + sp[j]);
  }

}

for(j in 1:2){
  Dec_HB[j] = mean(D_HB[,j]);
  Dec_ES[j] = mean(D_ES[,j]);
  Dec_P [j] = mean(D_P[,j]);
  Dec_M [j] = mean(D_M[,j]);
  Dec_DR[j] = mean(D_DR[,j]);
}

for(j in 1:3){
  Dec_HT[j] = mean(D_HT[,j]);
  Dec_R [j] = mean(D_R[,j]);
  Dec_S [j] = mean(D_S[,j]);
  Dec_MT[j] = mean(D_MT[,j]);
}

for (j in 1:11){
  Dec_SP[j] = mean(D_SP[,j]);
}


//// computing the log likelihood for model diagnostic and comparison

  for ( i in 1:N )(

     if ( decayed[i] == 0 ){
       log_lik[i] = gamma_lccdf(days_to_event[i] | mu[i] * phi, phi);
     }

     if ( decayed[i] == 1 ){
       log_lik[i]= gamma_lpdf(days_to_event[i] | mu[i] * phi, phi);
     }
  }
}
```