**Stan\_code\_Purse\_seine\_tuna\_geoGAMM**

Stan code used for the article*Evidence to inform spatiotemporal management of a western Pacific Ocean tuna purse seine fishery* follows for the combined tuna species geoGAMM model with hurdle lognormal likelihood, 2D smoothing spllne function with GP basis for the spatial effect with Matern covariance kernel and both vessel- and observer-specific random effects. The model uses the following software and packages:

* R version 4.4.1
* Stan version: 2.35.0
* Rstan version 2.32.6
* CmdStan version: 2.34.1
* cmdstanr version 0.7.1

functions {

/\* hurdle lognormal log-PDF of a single response

\* Args:

\* y: the response value

\* mu: mean parameter of the lognormal distribution

\* sigma: sd parameter of the lognormal distribution

\* hu: hurdle probability

\* Returns:

\* a scalar to be added to the log posterior

\*/

real hurdle\_lognormal\_lpdf(real y, real mu, real sigma, real hu) {

if (y == 0) {

return bernoulli\_lpmf(1 | hu);

} else {

return bernoulli\_lpmf(0 | hu) +

lognormal\_lpdf(y | mu, sigma);

}

}

/\* hurdle lognormal log-PDF of a single response

\* logit parameterization of the hurdle part

\* Args:

\* y: the response value

\* mu: mean parameter of the lognormal distribution

\* sigma: sd parameter of the lognormal distribution

\* hu: linear predictor for the hurdle part

\* Returns:

\* a scalar to be added to the log posterior

\*/

real hurdle\_lognormal\_logit\_lpdf(real y, real mu, real sigma, real hu) {

if (y == 0) {

return bernoulli\_logit\_lpmf(1 | hu);

} else {

return bernoulli\_logit\_lpmf(0 | hu) +

lognormal\_lpdf(y | mu, sigma);

}

}

// hurdle lognormal log-CCDF and log-CDF functions

real hurdle\_lognormal\_lccdf(real y, real mu, real sigma, real hu) {

return bernoulli\_lpmf(0 | hu) + lognormal\_lccdf(y | mu, sigma);

}

real hurdle\_lognormal\_lcdf(real y, real mu, real sigma, real hu) {

return log1m\_exp(hurdle\_lognormal\_lccdf(y | mu, sigma, hu));

}

}

data {

int<lower=1> N; // total number of observations

vector[N] Y; // response variable

int<lower=1> K; // number of population-level effects

matrix[N, K] X; // population-level design matrix

int<lower=1> Kc; // number of population-level effects after centering

// data for splines

int Ks; // number of linear effects

matrix[N, Ks] Xs; // design matrix for the linear effects

// data for spline 1

int nb\_1; // number of bases

array[nb\_1] int knots\_1; // number of knots

// basis function matrices

matrix[N, knots\_1[1]] Zs\_1\_1;

// data for spline 2

int nb\_2; // number of bases

array[nb\_2] int knots\_2; // number of knots

// basis function matrices

matrix[N, knots\_2[1]] Zs\_2\_1;

// data for spline 3

int nb\_3; // number of bases

array[nb\_3] int knots\_3; // number of knots

// basis function matrices

matrix[N, knots\_3[1]] Zs\_3\_1;

// data for spline 4

int nb\_4; // number of bases

array[nb\_4] int knots\_4; // number of knots

// basis function matrices

matrix[N, knots\_4[1]] Zs\_4\_1;

// data for spline 5

int nb\_5; // number of bases

array[nb\_5] int knots\_5; // number of knots

// basis function matrices

matrix[N, knots\_5[1]] Zs\_5\_1;

// data for spline 6

int nb\_6; // number of bases

array[nb\_6] int knots\_6; // number of knots

// basis function matrices

matrix[N, knots\_6[1]] Zs\_6\_1;

// data for splines

int Ks\_hu; // number of linear effects

matrix[N, Ks\_hu] Xs\_hu; // design matrix for the linear effects

// data for spline 1

int nb\_hu\_1; // number of bases

array[nb\_hu\_1] int knots\_hu\_1; // number of knots

// basis function matrices

matrix[N, knots\_hu\_1[1]] Zs\_hu\_1\_1;

// data for spline 2

int nb\_hu\_2; // number of bases

array[nb\_hu\_2] int knots\_hu\_2; // number of knots

// basis function matrices

matrix[N, knots\_hu\_2[1]] Zs\_hu\_2\_1;

// data for group-level effects of ID 1

int<lower=1> N\_1; // number of grouping levels

int<lower=1> M\_1; // number of coefficients per level

array[N] int<lower=1> J\_1; // grouping indicator per observation

// group-level predictor values

vector[N] Z\_1\_1;

// data for group-level effects of ID 2

int<lower=1> N\_2; // number of grouping levels

int<lower=1> M\_2; // number of coefficients per level

array[N] int<lower=1> J\_2; // grouping indicator per observation

// group-level predictor values

vector[N] Z\_2\_1;

int prior\_only; // should the likelihood be ignored?

}

transformed data {

matrix[N, Kc] Xc; // centered version of X without an intercept

vector[Kc] means\_X; // column means of X before centering

for (i in 2:K) {

means\_X[i - 1] = mean(X[, i]);

Xc[, i - 1] = X[, i] - means\_X[i - 1];

}

}

parameters {

vector[Kc] b; // regression coefficients

real Intercept; // temporary intercept for centered predictors

vector[Ks] bs; // unpenalized spline coefficients

// parameters for spline 1

// standardized penalized spline coefficients

vector[knots\_1[1]] zs\_1\_1;

vector<lower=0>[nb\_1] sds\_1; // SDs of penalized spline coefficients

// parameters for spline 2

// standardized penalized spline coefficients

vector[knots\_2[1]] zs\_2\_1;

vector<lower=0>[nb\_2] sds\_2; // SDs of penalized spline coefficients

// parameters for spline 3

// standardized penalized spline coefficients

vector[knots\_3[1]] zs\_3\_1;

vector<lower=0>[nb\_3] sds\_3; // SDs of penalized spline coefficients

// parameters for spline 4

// standardized penalized spline coefficients

vector[knots\_4[1]] zs\_4\_1;

vector<lower=0>[nb\_4] sds\_4; // SDs of penalized spline coefficients

// parameters for spline 5

// standardized penalized spline coefficients

vector[knots\_5[1]] zs\_5\_1;

vector<lower=0>[nb\_5] sds\_5; // SDs of penalized spline coefficients

// parameters for spline 6

// standardized penalized spline coefficients

vector[knots\_6[1]] zs\_6\_1;

vector<lower=0>[nb\_6] sds\_6; // SDs of penalized spline coefficients

real<lower=0> sigma; // dispersion parameter

real Intercept\_hu; // temporary intercept for centered predictors

vector[Ks\_hu] bs\_hu; // unpenalized spline coefficients

// parameters for spline 1

// standardized penalized spline coefficients

vector[knots\_hu\_1[1]] zs\_hu\_1\_1;

vector<lower=0>[nb\_hu\_1] sds\_hu\_1; // SDs of penalized spline coefficients

// parameters for spline 2

// standardized penalized spline coefficients

vector[knots\_hu\_2[1]] zs\_hu\_2\_1;

vector<lower=0>[nb\_hu\_2] sds\_hu\_2; // SDs of penalized spline coefficients

vector<lower=0>[M\_1] sd\_1; // group-level standard deviations

array[M\_1] vector[N\_1] z\_1; // standardized group-level effects

vector<lower=0>[M\_2] sd\_2; // group-level standard deviations

array[M\_2] vector[N\_2] z\_2; // standardized group-level effects

}

transformed parameters {

// penalized spline coefficients

vector[knots\_1[1]] s\_1\_1;

// penalized spline coefficients

vector[knots\_2[1]] s\_2\_1;

// penalized spline coefficients

vector[knots\_3[1]] s\_3\_1;

// penalized spline coefficients

vector[knots\_4[1]] s\_4\_1;

// penalized spline coefficients

vector[knots\_5[1]] s\_5\_1;

// penalized spline coefficients

vector[knots\_6[1]] s\_6\_1;

// penalized spline coefficients

vector[knots\_hu\_1[1]] s\_hu\_1\_1;

// penalized spline coefficients

vector[knots\_hu\_2[1]] s\_hu\_2\_1;

vector[N\_1] r\_1\_1; // actual group-level effects

vector[N\_2] r\_2\_1; // actual group-level effects

real lprior = 0; // prior contributions to the log posterior

// compute penalized spline coefficients

s\_1\_1 = sds\_1[1] \* zs\_1\_1;

// compute penalized spline coefficients

s\_2\_1 = sds\_2[1] \* zs\_2\_1;

// compute penalized spline coefficients

s\_3\_1 = sds\_3[1] \* zs\_3\_1;

// compute penalized spline coefficients

s\_4\_1 = sds\_4[1] \* zs\_4\_1;

// compute penalized spline coefficients

s\_5\_1 = sds\_5[1] \* zs\_5\_1;

// compute penalized spline coefficients

s\_6\_1 = sds\_6[1] \* zs\_6\_1;

// compute penalized spline coefficients

s\_hu\_1\_1 = sds\_hu\_1[1] \* zs\_hu\_1\_1;

// compute penalized spline coefficients

s\_hu\_2\_1 = sds\_hu\_2[1] \* zs\_hu\_2\_1;

r\_1\_1 = (sd\_1[1] \* (z\_1[1]));

r\_2\_1 = (sd\_2[1] \* (z\_2[1]));

lprior += student\_t\_lpdf(Intercept | 3, 2.3, 2.5);

lprior += student\_t\_lpdf(sds\_1 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += student\_t\_lpdf(sds\_2 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += student\_t\_lpdf(sds\_3 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += student\_t\_lpdf(sds\_4 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += student\_t\_lpdf(sds\_5 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += student\_t\_lpdf(sds\_6 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += student\_t\_lpdf(sigma | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += logistic\_lpdf(Intercept\_hu | 0, 1);

lprior += student\_t\_lpdf(sds\_hu\_1 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += student\_t\_lpdf(sds\_hu\_2 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += student\_t\_lpdf(sd\_1 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

lprior += student\_t\_lpdf(sd\_2 | 3, 0, 2.5)

- 1 \* student\_t\_lccdf(0 | 3, 0, 2.5);

}

model {

// likelihood including constants

if (!prior\_only) {

// initialize linear predictor term

vector[N] mu = rep\_vector(0.0, N);

// initialize linear predictor term

vector[N] hu = rep\_vector(0.0, N);

mu += Intercept + Xc \* b + Xs \* bs + Zs\_1\_1 \* s\_1\_1 + Zs\_2\_1 \* s\_2\_1 + Zs\_3\_1 \* s\_3\_1 + Zs\_4\_1 \* s\_4\_1 + Zs\_5\_1 \* s\_5\_1 + Zs\_6\_1 \* s\_6\_1;

hu += Intercept\_hu + Xs\_hu \* bs\_hu + Zs\_hu\_1\_1 \* s\_hu\_1\_1 + Zs\_hu\_2\_1 \* s\_hu\_2\_1;

for (n in 1:N) {

// add more terms to the linear predictor

mu[n] += r\_1\_1[J\_1[n]] \* Z\_1\_1[n] + r\_2\_1[J\_2[n]] \* Z\_2\_1[n];

}

for (n in 1:N) {

target += hurdle\_lognormal\_logit\_lpdf(Y[n] | mu[n], sigma, hu[n]);

}

}

// priors including constants

target += lprior;

target += std\_normal\_lpdf(zs\_1\_1);

target += std\_normal\_lpdf(zs\_2\_1);

target += std\_normal\_lpdf(zs\_3\_1);

target += std\_normal\_lpdf(zs\_4\_1);

target += std\_normal\_lpdf(zs\_5\_1);

target += std\_normal\_lpdf(zs\_6\_1);

target += std\_normal\_lpdf(zs\_hu\_1\_1);

target += std\_normal\_lpdf(zs\_hu\_2\_1);

target += std\_normal\_lpdf(z\_1[1]);

target += std\_normal\_lpdf(z\_2[1]);

}

generated quantities {

// actual population-level intercept

real b\_Intercept = Intercept - dot\_product(means\_X, b);

// actual population-level intercept

real b\_hu\_Intercept = Intercept\_hu;

}