```
data {
  // DATA FOR qPCR PART OF THE MODEL
  int Nplates; // number of PCR plates
  int Nobs_qpcr; // number of field observations for qPCR
  int NSamples_qpcr; //number of unique biological samples, overall
  int NstdSamples; //number of unique biological samples with known concentrations (standards)
  int plate_idx[Nobs_qpcr]; //index denoting which PCR plate each field sample is on
  int std_plate_idx[NstdSamples];//index denoting which PCR plate each standard sample is on
  real beta_std_curve_0_offset;
  vector[Nobs_qpcr] y_unk; //Ct for field observations
  int z_unk[Nobs_qpcr]; //indicator for field obs; z = 1 if a Ct was observed, z = 0 otherwise
  vector[NstdSamples] y_std; //Ct for standards
  int z_std[NstdSamples]; //indicator for standards; z = 1 if a Ct was observed, z = 0 otherwise
  vector[NstdSamples] known_concentration; //known concentration (copies/vol) in standards
  real stdCurvePrior_intercept[2]; // prior on the intercept of the std curves
  real stdCurvePrior_slope[2]; // prior on the slope of the std curves
  //Covariates and offsets
  vector[Nobs_qpcr] X_offset_tot; //log dilution and log volume offsets together in one vector.
  int N_station_depth;
  matrix[NSamples_qpcr,N_station_depth] X_station_depth_tube;// covariate design matrices
  matrix[Nobs_qpcr,N_station_depth] X_station_depth_obs; //covariate design matrices
  int N_bio_rep_RE;
  int N_bio_rep_param;
  int N_bio_rep_idx;
  int bio_rep_idx[N_bio_rep_idx] ;
  matrix[NSamples_qpcr,N_bio_rep_RE] X_bio_rep_tube;// covariate design matrices for unique samples.
  matrix[Nobs_qpcr,N_bio_rep_RE] X_bio_rep_obs;// covariate design matrices for observation
  vector[Nobs_qpcr] wash_idx;//design matrix for wash effect
  real wash_prior[2]; //priors for wash offset ~ N(wash_offset_prior[1],wash_offset_prior[2])
  //END DATA FOR qPCR
  // DATA FOR METABARCODING PART OF THE MODEL
  int N_species; // Number of species in data
  int N_obs_mb_samp; // Number of observed samples, also the number of groups for qPCR samps to link to
  int N_obs_mb_samp_small; // Number of observed samples for individual sites.
  int N_obs_mock; // Number of observed mock samples
  // Observed data of community matrices
  int sample_data[N_obs_mb_samp,N_species];
  // Observed data of mock community matrices
  int mock_data[N_obs_mock,N_species];
  // True proportions for mock community
  matrix[N_obs_mock,N_species] alr_mock_true_prop ;
 // matrix[N_obs_mock_small,N_species] alr_mock_true_prop_small ;
  // Design matrices: field samples
  int N_b_samp_col; // Number of samples
  matrix[N_obs_mb_samp,N_b_samp_col] model_matrix_b_samp; // all samples design matrix
  matrix[N_obs_mb_samp_small,N_b_samp_col] model_matrix_b_samp_small; // all samples with replicates co
  vector[N_obs_mb_samp] model_vector_a_samp; // Npcr cycles for each sample, replicates included
  vector[N_obs_mb_samp_small] model_vector_a_samp_small; // Npcr cycles without replicates
  // Design matrices: mock community samples
  vector[N_obs_mock] model_vector_a_mock;
  // Identify a reference species for each observation (most abundant species in each sampl)
  int ref_sp_idx[N_obs_mb_samp];
  // Priors
```

```
  real alpha_prior[2]; // Parameters of normal distribution for prior on alphas
  real tau_prior[2]; // Parameters of gamma distribution for prior on tau (observation precision)
  // END DATA FOR METABARCODING
  // DATA FOR LINKING QM AND QPCR
  int N_mb_link; //How many qpCR samples have a match in a MB sample
  int mb_link_idx[N_mb_link]; // index: which qpcr samples (plateSample_idx) does each MB sample corres
  int mb_link_sp_idx; // the index for the species linking QM to qPCR (usually hake)
  int tube_link_idx[N_obs_mb_samp]; //index linking observations to unique biological samples
}
transformed data {
  vector[NstdSamples] log_known_conc; // log known concentration of qPCR standards
  matrix[N_obs_mb_samp,N_b_samp_col+1] model_matrix_samp; // QM design matrix (samples by species), whe
  log_known_conc = log(known_concentration);
  model_matrix_samp = append_col(model_matrix_b_samp, model_vector_a_samp); //extend MB design matrix t
  // The model_matrix will be multiplied by the relative abundances (log(conc) relative to qpcr referen
}
parameters {
  // for QM part
  //real<lower=0> tau_base; // single overdispersion sd for multinomial.
  //real<lower=0> tau; // single overdispersion sd for multinomial.
  vector[N_species-1] alpha_raw;
  // vector[N_obs_mb_samp] eta_samp_raw[N_species-1]; //overdispersion
  // vector[N_obs_mock] eta_mock_raw[N_species-1]; //overdispersion
  // for qPCR part
  real mean_hake;
  vector[Nplates] beta_std_curve_0; // intercept of standard curve
  vector<lower= stdCurvePrior_slope[1]>[Nplates] beta_std_curve_1; // slope of standard curve
  real gamma_0; //intercept to scale variance of standard curves w the mean
  real<upper=0> gamma_1; //slopes to scale variance of standards curves w the mean
   real<upper= 1.854586> phi_0; // this bound is the logistic transform of dpois(0,2)... which is the p
   real<lower=0> phi_1;
  vector[N_station_depth] log_D_station_depth; // log DNA concentration in field samples in each tube
  real<lower=0> log_D_sigma;
  vector[N_bio_rep_param] bio_rep_param; // log DNA concentration in field samples
  real<upper=0> wash_effect; //estimate of the EtOH wash effect
  real<lower=0> tau_bio_rep; # random effect for biological replicates.
  //for linking
  matrix[N_obs_mb_samp,(N_species-1)] log_D_raw; // estimated true DNA concentration by sample
}
transformed parameters {
  // for qPCR part
  vector[Nobs_qpcr] Ct; // estimated Ct for all unknown qPCR samples
  vector[Nobs_qpcr] unk_conc_qpcr; //log DNA concentration in field samples observed in qPCR after adju
  vector[NSamples_qpcr] log_D_station_depth_tube; //log DNA concentration in field samples (tubes)
  vector[NstdSamples] Ct_std; //estimated Ct for standards
  vector[NstdSamples] sigma_std; // SD of Ct values, standards
  vector[NstdSamples] logit_theta_std; // Bernoulli param, probability of amplification, standards
  vector[Nobs_qpcr] sigma_samp; //SD of Ct values, field samples
  vector[Nobs_qpcr] logit_theta_samp; //Probability of amplification, field samples
  vector[N_bio_rep_RE] bio_rep_RE; // log DNA concentration in field samples
  // for QM part
  vector[N_species] alpha; // vector of coefficients (log-efficiencies relative to reference taxon)
  // vector[N_obs_mb_samp] eta_samp[N_species]; // overdispersion coefficients
```

```
vector[N_obs_mock] eta_mock[N_species]; // overdispersion coefficients
matrix[N_obs_mb_samp,N_species] logit_val_samp;
matrix[N_obs_mock,N_species] logit_val_mock;
// matrix[N_obs_mb_samp,N_species] mu_samp; // estimates of read counts, in log space
// matrix[N_obs_mock,N_species] mu_mock; // estimates of read counts, in log space
// for linking
matrix[N_obs_mb_samp,N_species] log_D; // estimated true copy numbers by sample, including the link s
// qPCR standard curves (vectorized)
Ct_std = beta_std_curve_0_offset+beta_std_curve_0[std_plate_idx] +
                          beta_std_curve_1[std_plate_idx] .* log_known_conc;
sigma_std = exp(gamma_0 + gamma_1 .* log_known_conc);
logit_theta_std = phi_0 + phi_1 .* exp(log_known_conc);
// for(i in 1:NstdSamples){
//   // Ct_std[i] = beta_std_curve_0_offset+beta_std_curve_0[std_plate_idx[i]] +
//   //                            beta_std_curve_1[std_plate_idx[i]] * log_known_conc[i];
//   if(theta_std[i]==1){theta_std[i]= 1 - 1e-10; }
// }
// qPCR unknowns
  {// locals for making sum to 0 random effects.
    int count_tot;
    int count_par;
    real bio_rep_sum;
  // random effect of biological replicate
  // This does depend on the stations and tubes being in order from small to large.
  count_tot = 0;
  count_par = 0;
  for(j in 1:N_bio_rep_idx){
    bio_rep_sum = 0 ;
    for(k in 1:bio_rep_idx[j]){
      count_tot = count_tot + 1;
      if(k < bio_rep_idx[j]){
        count_par = count_par + 1;
        bio_rep_RE[count_tot] = bio_rep_param[count_par] * tau_bio_rep ;
        bio_rep_sum = bio_rep_sum + bio_rep_RE[count_tot];
      }else if(bio_rep_idx[j]==1){
        bio_rep_RE[count_tot] = 0 ;
      }else{
        bio_rep_RE[count_tot] = -bio_rep_sum;
      }
        } // end k loop
      } // end j loop
    } // end local variables.
/// THIS IS THE LATENT STATE THAT WILL BE NEEDED TO CONNECT TO THE MB DATA
log_D_station_depth_tube = mean_hake + X_station_depth_tube * log_D_station_depth +
                         X_bio_rep_tube * bio_rep_RE ;
/// THIS IS THE LATENT STATE CONNECTS TO THE QPCR OBSERVATIONS
unk_conc_qpcr = mean_hake + X_station_depth_obs * log_D_station_depth +
                    X_bio_rep_obs * bio_rep_RE +
                    wash_idx * wash_effect +
                    X_offset_tot ;
// Vectorized predictions
Ct = (beta_std_curve_0_offset + beta_std_curve_0[plate_idx]) + beta_std_curve_1[plate_idx].*unk_conc_
sigma_samp = exp(gamma_0 + gamma_1 .* unk_conc_qpcr );
```

```
  logit_theta_samp = phi_0 + phi_1 *exp(unk_conc_qpcr);
  // for(i in 1:Nobs_qpcr){
  //   //Ct[i] = 36+beta_std_curve_0[plate_idx[i]]+beta_std_curve_1[plate_idx[i]]*unk_conc_qpcr[i];
  //   if(theta_samp[i]==1){theta_samp[i]= 1 - 1e-10; }
  // }
  //Link to QM
  for(i in 1:N_species){
    for(j in 1:N_obs_mb_samp){
      if(i==mb_link_sp_idx){ // if index is equal to link species, fill in qpcr estimate
        log_D[j,i] = log_D_station_depth_tube[tube_link_idx[j]];
      }else{ // otherwise, fill from log_D_raw
        if(i<mb_link_sp_idx){
          log_D[j,i] = log_D_raw[j,i];
        }else{
          log_D[j,i] = log_D_raw[j,(i-1)];
        }
      }
    }
  }
  // QM MODEL PIECES
  // Fixed effects components
  alpha[1:(N_species-1)] = alpha_prior[1] + alpha_raw * alpha_prior[2];
        // non-centered param beta ~ normal(alpha_prior[1], alpha_prior[2])
  alpha[N_species] = 0; // final species is zero (reference species)
  // tau = rep_vector(tau_base,N_species-1);
  // eta_mock[N_species] = rep_vector(0.0,N_obs_mock); // final species is zero (reference species)
      // eta_samp[N_species] = rep_vector(0.0,N_obs_mb_samp); // final species is zero (reference speci
  // random effects vector of vectors
  // for (l in 1:(N_species-1)) {
  //   eta_mock[l] = eta_mock_raw[l] * tau ; // non-centered param eta_mock ~ normal(0,tau)
  //   // eta_samp[l] = eta_samp_raw[l] * tau ; // non-centered param eta_samp ~ normal(0,tau)
  // }
// from qPCR estimates, alphas, and etas we can calculate sample-specific mu
  // Make a vector for the reference species D and for alpha
 {// local variables for making reference species vectors
      vector[N_obs_mb_samp] log_D_ref;
      vector[N_obs_mb_samp] alpha_ref;
  for(i in 1:N_obs_mb_samp){
     log_D_ref[i] = log_D[i,ref_sp_idx[i]];
     alpha_ref[i] = alpha[ref_sp_idx[i]];
  }
    // If you wanted variable reference species in the mocks, this is where you would do it.
  // for(i in 1:N_obs_mock){
  //    log_D_ref[i] = log_D[i,ref_sp_idx[i]];
  //    alpha_ref[i] = alpha[ref_sp_idx[i]];
  // }
  for (n in 1:N_species) {
    logit_val_samp[,n] = (log_D[,n] - log_D_ref) + model_vector_a_samp.*(alpha[n] - alpha_ref);
                          //model_matrix_samp * append_row((log_D[,n] - log_D[,ref_sp_idx]),alpha[n] -
                            //+eta_samp[n];
    logit_val_mock[,n] = alr_mock_true_prop[,n] +
                               model_vector_a_mock * (alpha[n]) ;
                               // eta_mock[n];
```

```
  }
  // for(m in 1:N_obs_mb_samp){
  //   prob_samp_t[,m] = softmax(transpose(logit_val_samp[m,])); // proportion of each taxon in field s
  // }
  // for(m in 1:N_obs_mock){
  //    prob_mock_t[,m] = softmax(transpose(logit_val_mock[m,])); // proportion of each taxon in mocks
  // }
  // for (n in 1:N_species) {
  //     mu_samp[,n] = transpose(prob_samp_t)[,n] ;
  //     mu_mock[,n] = transpose(prob_mock_t)[,n] ;
  //   // if(n==1){print("log_prob 1 ",log_prob);}
  // }
   } // end local variables
    // print("MU_SAMP_row",mu_samp[1,]);
    // print("SUM_MU_SAMP",sum(mu_samp[1,]));
    //
    // print("MU_SAMP_col",mu_samp[,1]);
    // print("SUM_MU_SAMP_col",sum(mu_samp[,1]));
}
model{
  // print("HERE1",target());
  // qPCR part
  z_std ~ bernoulli_logit(logit_theta_std);
  for(i in 1:NstdSamples){
    if(z_std[i]==1){
      y_std[i] ~ normal(Ct_std[i],sigma_std[i]);
    }
  }
  // print("HERE2",target());
  z_unk ~ bernoulli_logit(logit_theta_samp);
  // print(max(theta_samp)," IIIII ", min(theta_samp));
  // print("HERE2.5",target());
  for(i in 1:Nobs_qpcr){
     if (z_unk[i]==1){ //if Ct observed, then compute likelihood
        y_unk[i] ~ normal(Ct[i], sigma_samp[i]);
      }
    }
  //beta standard curve params
  beta_std_curve_0 ~ normal(stdCurvePrior_intercept[1]-beta_std_curve_0_offset, stdCurvePrior_intercept
  beta_std_curve_1 ~ normal(stdCurvePrior_slope[1], stdCurvePrior_slope[2]);
  //gamma params for scaling variance on the standards
  gamma_1 ~ normal(0,1);
  gamma_0 ~ normal(0,1);
  bio_rep_param ~ std_normal();
  tau_bio_rep ~ normal(0,0.2);
  for(i in 1:(N_species-1)){
    // ONLY set a prior for the species that ARE NOT the qPCR link species (hake)
    // The values for the link species will come from the qPCR part of the joint model
    // (which will use prior information from envir_concentration, above)
    //if(i!=mb_link_sp_idx){
      log_D_raw[,i] ~ normal(3,10);
    //}
  }
```

```
  phi_0 ~ normal(1.854586,0.3); //assuming Poisson from bottles to replicates (pipetting)
  phi_1 ~ normal(1, 1);
  wash_effect ~ normal(wash_prior[1],wash_prior[2]) ;
    mean_hake ~normal(2,8);
    log_D_sigma ~ normal(0,3);
    log_D_station_depth ~ normal(0,log_D_sigma); //log scale
  // print("1:",target());
  // QM Likelihoods
  for(i in 1:N_obs_mock){
    mock_data[i,] ~ multinomial_logit(transpose(logit_val_mock[i,])); // Multinomial sampling of mu (pr
  }
  // print("2:",target());
  for(i in 1:N_obs_mb_samp){
    sample_data[i,] ~ multinomial_logit(transpose(logit_val_samp[i,])); // Multinomial sampling of mu (
  }
 //  print("3:",target());
  // Priors
  // for(i in 1:(N_species-1)){
  //   // eta_samp_raw[i] ~ std_normal(); // N(0,tau)
  //   eta_mock_raw[i] ~ std_normal(); // N(0,tau)
  // }
  // tau ~ normal(tau_prior[1],tau_prior[2]);
}
```