

# Appendix S1: Simulation Study using BUMMER Simulated data

## S1.1 Simulation Study

In this appendix, we simulated data using the BUMMER functional response model that assumes each species has a marginal symmetric, unimodal response. We simulated the log-scale random effect  $\alpha_{ij} = \exp\left(a_j - \frac{(b_j - x_i)^2}{c_j^2}\right)$  for  $N = 500$  observations where the covariate is observed and  $\tilde{N} = 200$  observations where the covariate is unobserved. The simulated data uses  $d = 8$  species. The parameters used in simulation are simulated from the following distributions:  $a_j \sim N(0, 1)$ ,  $b_j \sim N(0, 1)$ , and  $c_j \sim \text{gamma}(5, 2)$ . The simulation of the data using the `sim_compositional_data()` function in the `BayesComposition` package for the BUMMER model is shown below.

```
set.seed(101)
library(BayesComposition)
library(knitr)
library(here)
library(ggplot2)
library(GGally)
library(analogue)
library(rioja)
library(xtable)

## change these for your file structure
## output directory for MCMC output
save_directory <- "~/Google Drive/mvgp-test/"
## directory for MCMC monitoring output
progress_directory <- "./mvgp-test/"

## Function from BayesComposition to simulate data from BUMMER model
N <- 500
N_pred <- 200
d <- 8

dat <- sim_compositional_data(
  N = N,
  N_pred = N_pred,
  d = d,
  likelihood = "dirichlet-multinomial",
  function_type = "bummer")

p_alpha <- t(sapply(1:N, function(i) dat$alpha[i, ] / sum(dat$alpha[i, ])))
y_prop <- t(sapply(1:N, function(i) dat$y[i, ] / sum(dat$y[i, ])))

simPlotData <- data.frame(
  species = as.factor(rep(1:d, each = N)),
  count = c(y_prop),
  depth = rep(dat$X, times = d),
```

```

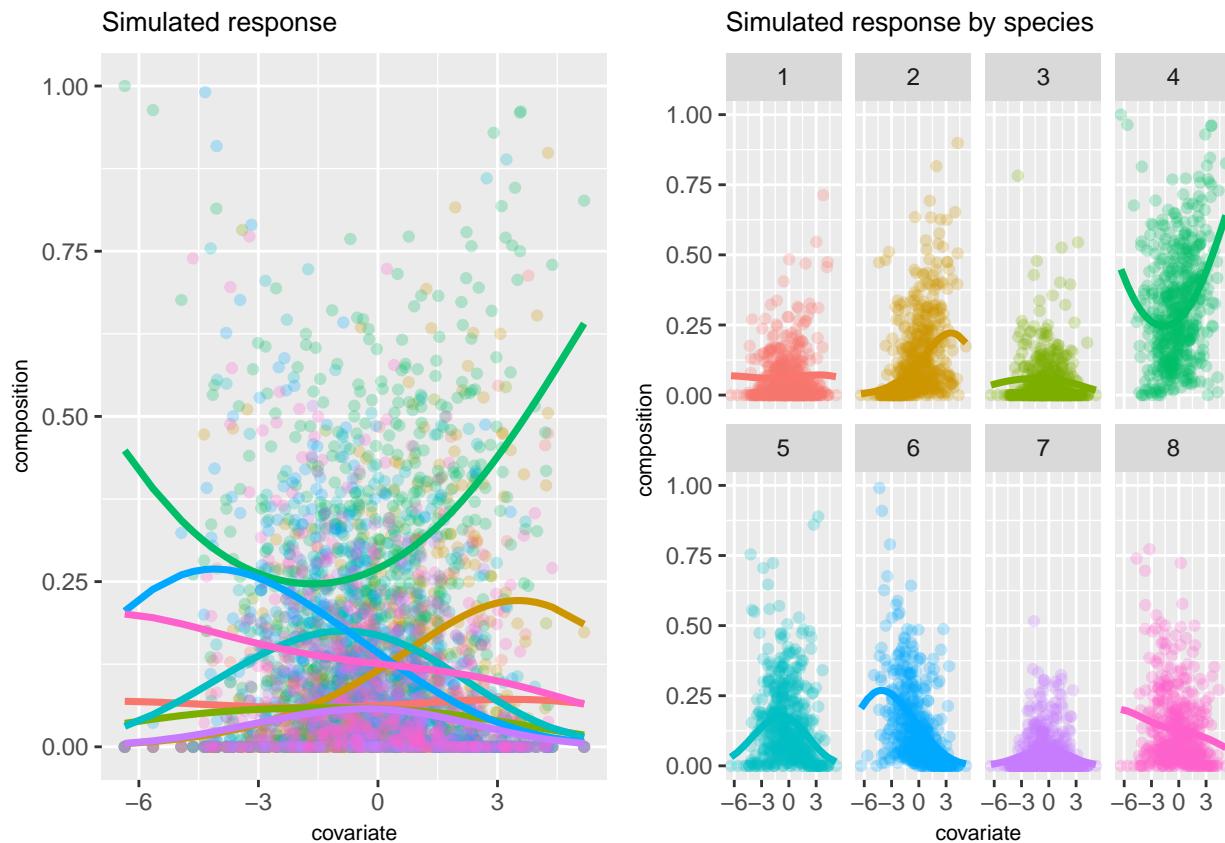
alpha    = c(p_alpha))

gsim1 <- ggplot(simPlotData, aes(x=depth, y=count, color=species,
                                  group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
        title = element_text(size=8)) +
  ggtitle("Simulated response") +
  labs(x = "covariate", y="composition") +
  geom_line(aes(x=depth, y=alpha, col = species), simPlotData, lwd=1.25)

gsim2 <- ggplot(simPlotData, aes(x=depth, y=count, color=species,
                                  group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
        title = element_text(size=8)) +
  ggtitle("Simulated response by species") +
  labs(x = "covariate", y="composition") +
  geom_line(aes(x=depth, y=alpha, col = species), simPlotData, lwd=1.25) +
  facet_wrap(~ species, ncol = 4)

## generate the plot
multiplot(gsim1, gsim2, cols=2)

```



## S1.2 Fitting the MVGP model

To fit the MVGP model, we consider  $n^* = 30$  evenly spaced knots over a range larger than the observed data  $X$  to allow for potential predictions outside the range of the prediction. We have not found the inference to show much sensitivity in practical applications to the number of knots. The Gaussian process kernel uses an exponential covariance function; other covariance functions currently included in the software include a Gaussian covariance function. We fit the model for 200,000 MCMC iterations, where the first 50,000 iterations include adaptive tuning of the Metropolis-Hastings proposals and are used as a burn-in [Roberts and Rosenthal, 2009]. We fit 4 chains in parallel, keeping every 150th iteration to reduce memory and storage costs resulting in 4,000 posterior samples.

```
## generate a sequence of knots that covers ranges outside
##      the observed values
n_knots <- 30
X_knots <- seq(
  min(dat$X, na.rm=TRUE) - 1.25 * sd(dat$X, na.rm=TRUE),
  max(dat$X, na.rm=TRUE) + 1.25 * sd(dat$X, na.rm=TRUE),
  length = n_knots)

## define the model parameters
params <- list(
  n_adapt          = 50000,
  n_mcmc           = 150000,
  n_thin            = 150,
  correlation_function = "exponential",
  likelihood        = "dirichlet-multinomial",
  function_type     = "gaussian-process",
  multiplicative_correlation = TRUE,
  additive_correlation = FALSE,
  n_chains          = 4,
  n_cores            = 4,
  n_knots           = n_knots,
  X_knots           = X_knots)

if (file.exists(paste0(save_directory, "sim-bummer-fit-mvgp.RData"))) {
  load(paste0(save_directory, "sim-bummer-fit-mvgp.RData"))
} else {
  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                  = dat$y,
    X                  = dat$X,
    params             = params,
    progress_directory = progress_directory,
    progress_file      = "sim-bummer-fit-mvgp.txt")
  ## save the output file
  save(out, file = paste0(save_directory, "sim-bummer-fit-mvgp.RData"))
}
```

### S1.2.1 MVGP convergence diagnostics

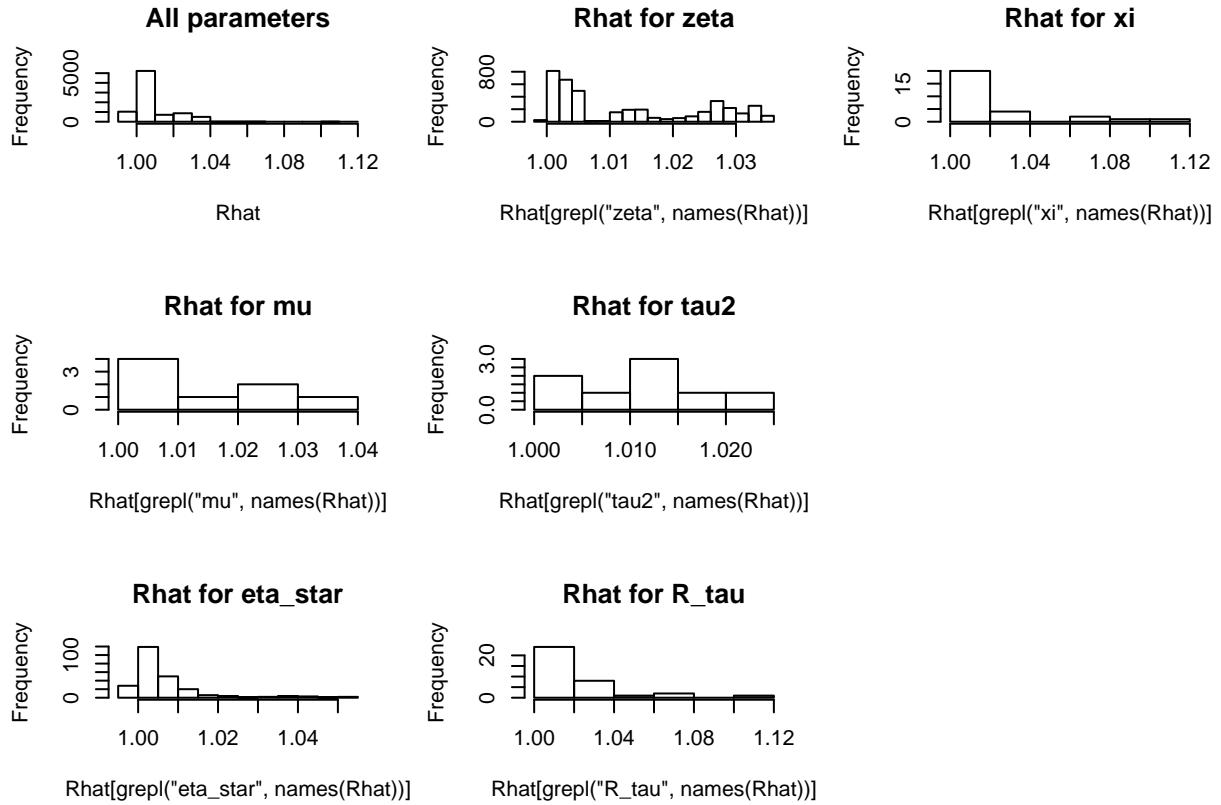
After fitting the model, the posterior samples are generated and MCMC convergence is assessed using the Gelman-Rubin  $\hat{R}$  statistic. We see the distribution of  $\hat{R}$  statistics is close to 1 with the largest  $\hat{R} = 1.1141744 \lesssim 1.1$  at an acceptable value (Gelman et al. [2013], pp.287).

```
## extract posterior samples
samples <- extract_compositional_samples(out)
mu_post <- samples$mu
eta_star_post <- samples$eta_star
zeta_post <- samples$zeta
Omega_post <- samples$Omega
phi_post <- samples$phi
tau2_post <- samples$tau2
R_post <- samples$R
R_tau_post <- samples$R_tau
xi_post <- samples$xi

n_samples <- nrow(mu_post)

## Calculate Gelman-Rubin convergence statistic
Rhat <- make_gelman_rubin(out)
layout(matrix(1:9, 3, 3))
hist(Rhat, main="All parameters")
hist(Rhat[grep("mu", names(Rhat))], main = "Rhat for mu")
hist(Rhat[grep("eta_star", names(Rhat))], main = "Rhat for eta_star")
hist(Rhat[grep("zeta", names(Rhat))], main = "Rhat for zeta")
hist(Rhat[grep("tau2", names(Rhat))], main = "Rhat for tau2")
hist(Rhat[grep("R_tau", names(Rhat))], main = "Rhat for R_tau")
hist(Rhat[grep("xi", names(Rhat))], main = "Rhat for xi")
Rhat[!is.finite(Rhat)] <- NA
max(unlist(na.omit(Rhat)))

## [1] 1.114174
```



### S1.2.2 MVGP functional response estimation

To compare how well the model reconstructs the unobserved latent functional relationships between the covariate and the observations, we plot both the simulated and fitted functional responses below. These show that the model is accurately estimating the latent functional responses, although the estimated functional responses are slightly rougher due to the MVGP model being a more complicated functional form than the BUMMER simulated data.

```
alpha_post_mean <- exp(t(apply(mu_post, 2, mean) +
                           t(apply(zeta_post, c(2, 3), mean))))
```

```
p_alpha_post_mean <- t(sapply(1:N, function(i) {
  alpha_post_mean[i, ] / sum(alpha_post_mean[i, ])))
})
```

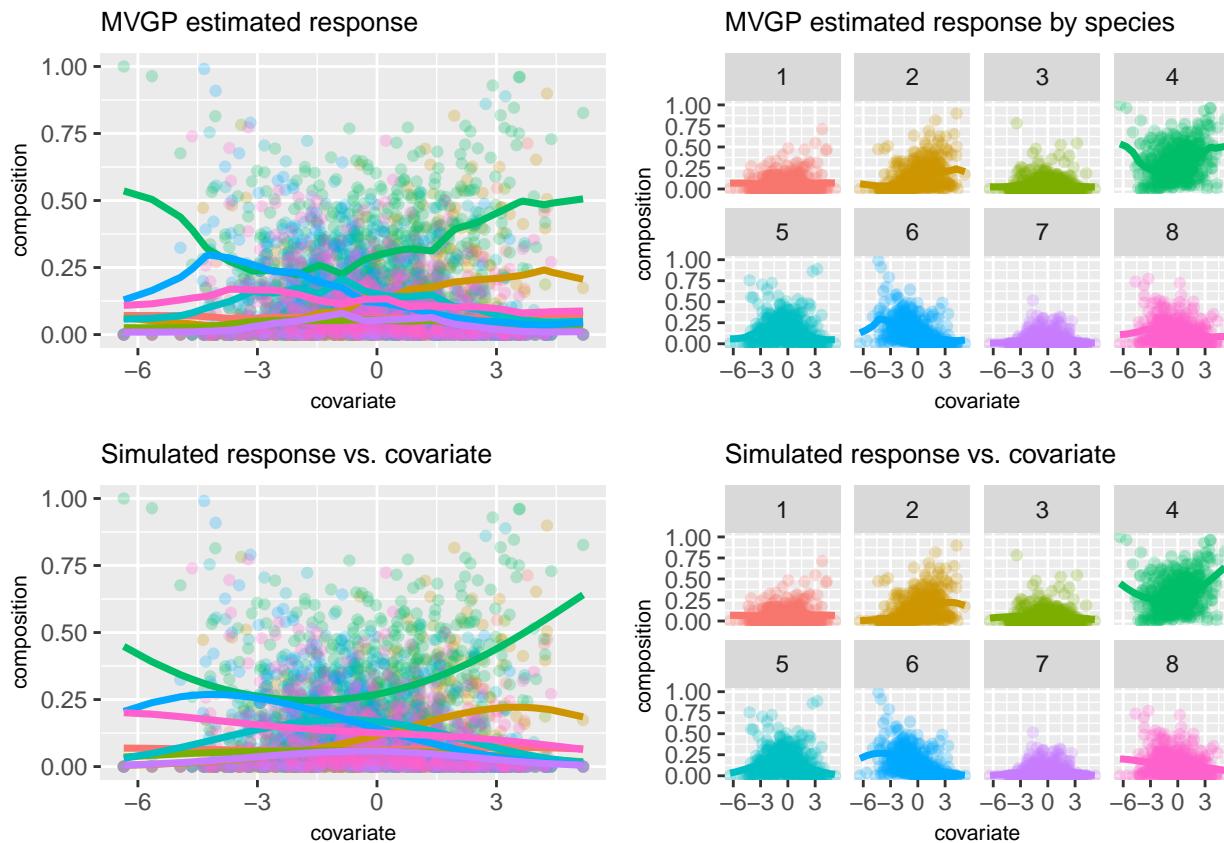
```
fitPlotData <- data.frame(species=as.factor(rep(1:d, each=N)),
                            count=c(y_prop),
                            depth=rep(dat$X, times=d),
                            alpha=c(p_alpha_post_mean))
```

```
g1_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
        title = element_text(size=8)) +
  ggtitle("MVGP estimated response") +
  labs(x = "covariate", y="composition") +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25)
```

```

g2_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
        title = element_text(size=8)) +
  ggtitle("MVGP estimated response by species") +
  labs(x = "covariate", y="composition") +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25) +
  facet_wrap(~ species, ncol = 4)
multiplot(g1_post, g2_post, gsim1, gsim2, cols=2)

```



### S1.2.3 Generate MVGP predictions

Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved covariates using the posterior samples estimated from the calibration model. These estimates can be generated either jointly with the calibration model or estimated using a two-stage approach as presented here. We have found, there is little practical difference in model performance between joint and two-stage estimation and follow the convention in the transfer function literature of using two-stage estimation.

```

## Note, there might be some initial warnings about ESS shrunk to the
## current position for the first few iterations. This is not a
## major concern as long as the warnings don't continue.
if (file.exists(paste0(save_directory, "sim-bummer-predict-mvgp.RData"))) {
  load(paste0(save_directory, "sim-bummer-predict-mvgp.RData"))
} else {
  pred <- predict_compositional_data(
    dat$y_pred,

```

```

dat$X,
params,
samples,
progress_directory,
"sim-bummer-predict-mvgp.txt")
save(pred, file = paste0(save_directory, "sim-bummer-predict-mvgp.RData"))
}

```

#### S1.2.4 Plot MVGP predictions

The predictions for 25 randomly chosen missing covariate observations are shown below. The simulated covariate values are shown as red dots and the posterior distribution shown as a violin plot where the width of the violin is proportional to the posterior density. The posterior mean is shown in blue and the posterior median is shown in orange. In general, the posterior distributions are doing a good job of estimating the unobserved covariate. Note there is multi-modal posterior mass evident in the posterior predictive distributions.

```

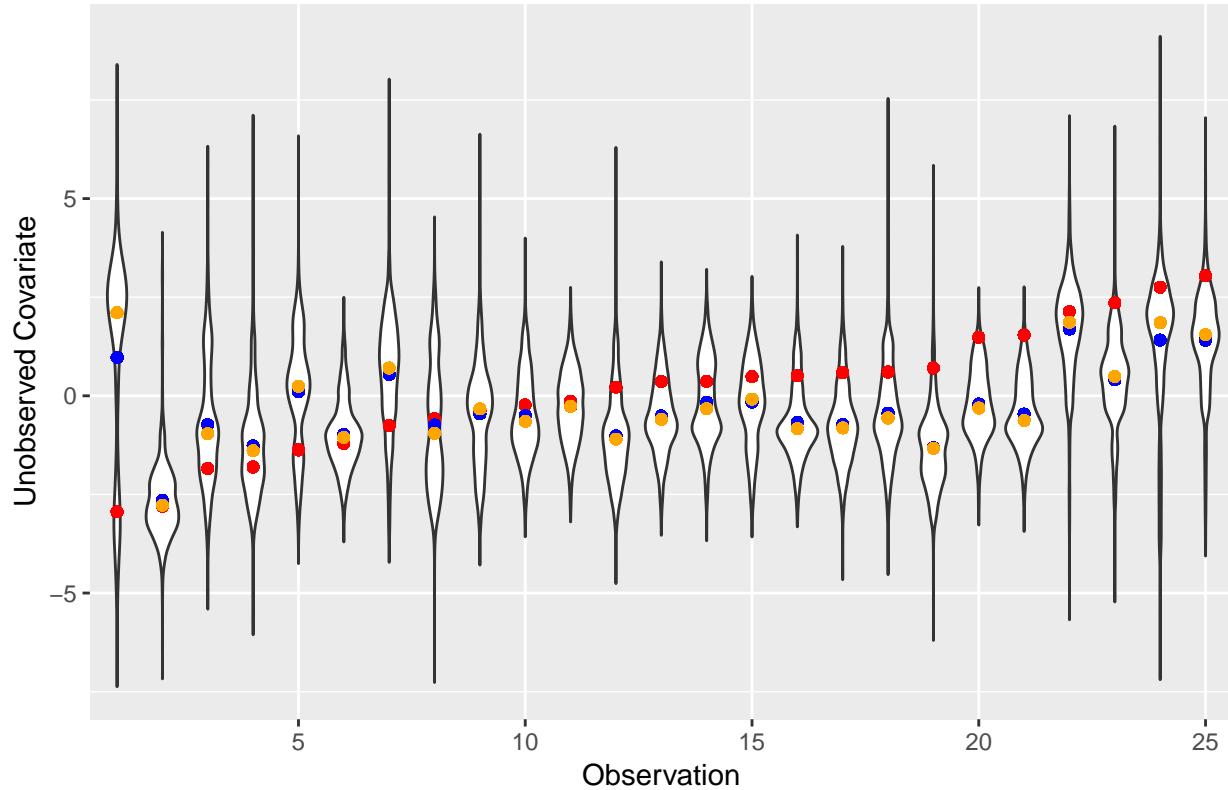
## sorted to increasing values for ease of display
idx <- order(dat$X_pred)
## randomly choose 25 observations to display
n_plot <- 25
plot_idx <- sort(sample(idx, n_plot))
n_samples <- length(pred$X[, 1])

sim_df <- data.frame(
  Covariate = c(pred$X[, idx[plot_idx]]),
  Mean      = rep(apply(pred$X[, idx[plot_idx]], 2, mean), each = n_samples),
  Median    = rep(apply(pred$X[, idx[plot_idx]], 2, median), each = n_samples),
  Observation = factor(rep(1:n_plot, each = n_samples )),
  truth      = rep(dat$X_pred[idx[plot_idx]], each = n_samples))

ggplot(sim_df, aes(Observation, Covariate)) +
  geom_violin(position="identity") +
  geom_point(aes(Observation, truth), color="red") +
  geom_point(aes(Observation, Mean), color="blue") +
  geom_point(aes(Observation, Median), color="orange") +
  scale_x_discrete(breaks=seq(5, n_plot, 5)) +
  labs(x="Observation", y="Unobserved Covariate") +
  ggtitle("MVGP prediction for BUMMER simulated data")

```

## MVGP prediction for BUMMER simulated data



### S1.3 Fitting the GAM model

The GAM model is a simplification to the full MVGP model. The `BayesComposition` package has an option for fitting the GAM model. We fit the model for 200,000 MCMC iterations, discarding the first 50,000 iterations with adaptive Metropolis-Hastings proposals as burn in and thinning every 150 iterations. The model is run for 4 chains resulting in 4000 posterior samples. The GAM model uses a cubic B-spline basis with 6 degrees of freedom and hierarchical pooling priors on the B-spline coefficients  $\beta$  using a log-link function.

```
## define the model parameters for the GAM model
params <- list(
  n_adapt          = 50000,
  n_mcmc           = 150000,
  n_thin            = 150,
  likelihood        = "dirichlet-multinomial",
  function_type     = "basis",
  multiplicative_correlation = FALSE,
  additive_correlation = FALSE,
  n_chains          = 4,
  n_cores            = 4,
  df                = 6)

if (file.exists(paste0(save_directory, "sim-bummer-fit-gam.RData"))) {
  load(paste0(save_directory, "sim-bummer-fit-gam.RData"))
} else {
  ## potentially long running MCMC code
```

```

out_gam <- fit_compositional_data(
  y           = dat$y,
  X           = dat$X,
  params      = params,
  progress_directory = progress_directory,
  progress_file    = "sim-bummer-fit-gam.txt")
## save the file
save(out_gam, file = paste0(save_directory, "sim-bummer-fit-gam.RData"))
}

```

### S1.3.1 GAM convergence diagnostics

After fitting the model, the posterior samples are generated and MCMC convergence is assessed using the Gelman-Rubin  $\hat{R}$  statistic. We see the distribution of  $\hat{R}$  statistics is close to 1 with the largest  $\hat{R} = 1.0344391 \leq 1.1$  at an acceptable value (Gelman et al. [2013], pp.287).

```

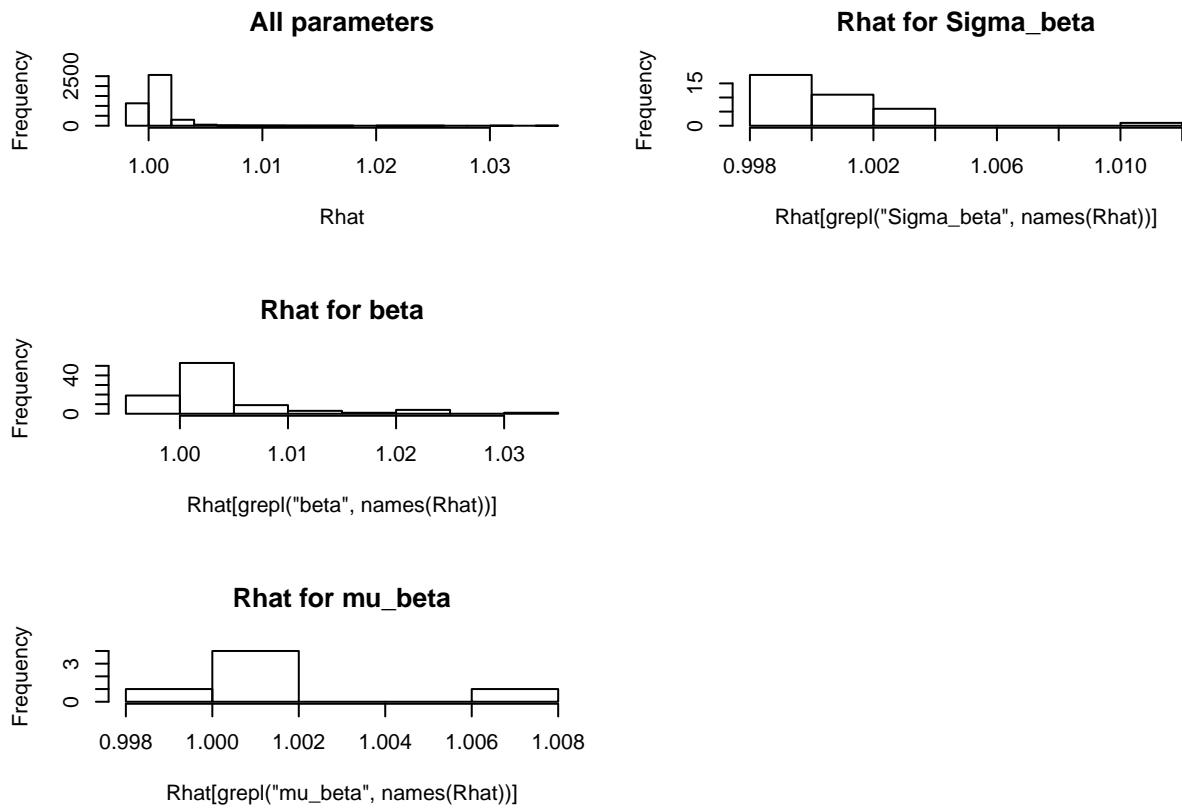
## extract posterior samples
samples_gam <- extract_compositional_samples(out_gam)
alpha_post <- samples_gam$alpha
beta_post <- samples_gam$beta
mu_beta_post <- samples_gam$mu_beta
Sigma_beta_post <- samples_gam$Sigma_beta

n_samples <- nrow(alpha_post)

## Calculate Gelman-Rubin convergence statistic
Rhat <- make_gelman_rubin(out_gam)
layout(matrix(1:6, 3, 2))
hist(Rhat, main="All parameters")
hist(Rhat[grep("beta", names(Rhat))], main = "Rhat for beta")
hist(Rhat[grep("mu_beta", names(Rhat))], main = "Rhat for mu_beta")
hist(Rhat[grep("Sigma_beta", names(Rhat))], main = "Rhat for Sigma_beta")
Rhat[!is.finite(Rhat)] <- NA
max(unlist(na.omit(Rhat)))

## [1] 1.034439

```



### S1.3.2 GAM functional response estimation

We compare how well the GAM model reconstructs the simulated functional relationships below. The GAM model estimates the functional form of the simulated BUMMER model well.

```
alpha_post_mean <- apply(alpha_post, c(2, 3), mean)

p_alpha_post_mean <- t(sapply(1:N, function(i) {
  alpha_post_mean[i, ] / sum(alpha_post_mean[i, ])
}))

fitPlotData <- data.frame(species=as.factor(rep(1:d, each=N)),
                           count=c(y_prop),
                           depth=rep(dat$X, times=d),
                           alpha=c(p_alpha_post_mean))

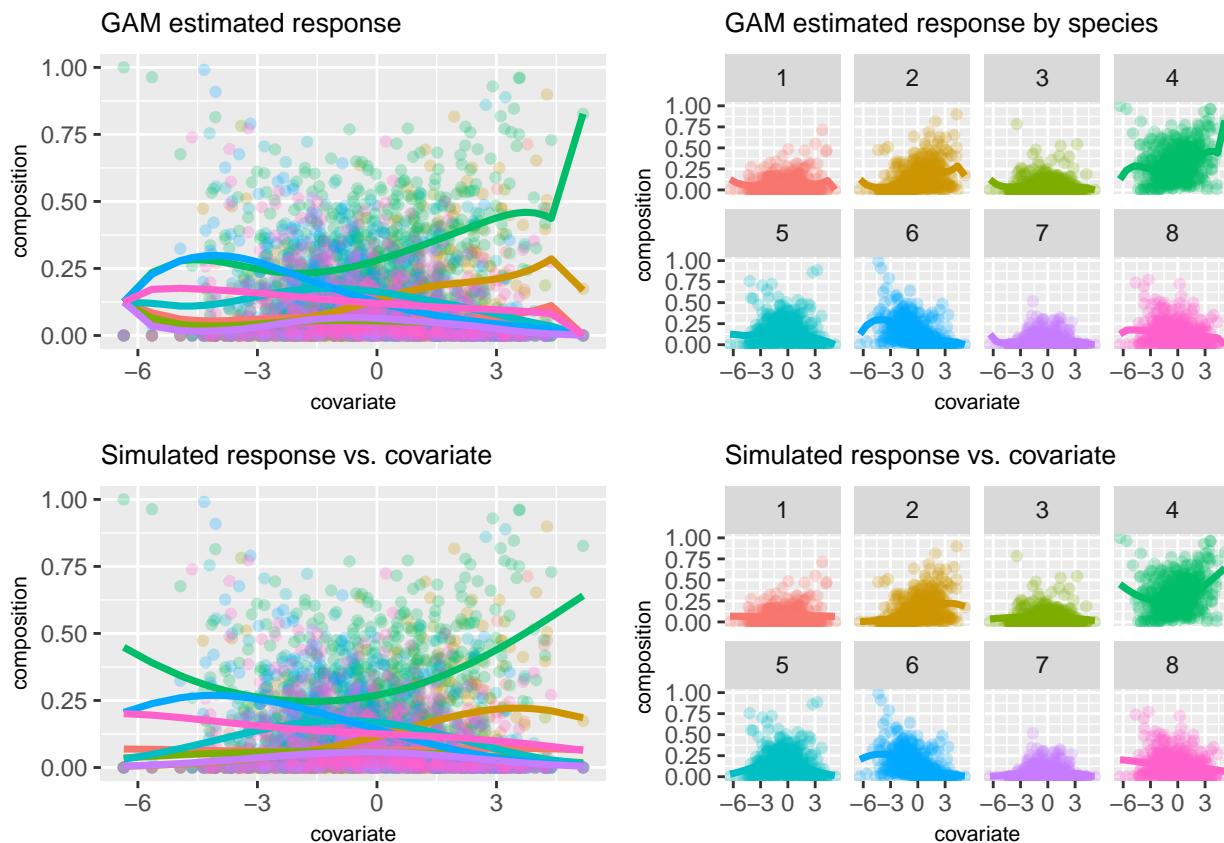
g1_gam_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
        title = element_text(size=8)) +
  ggtitle("GAM estimated response") +
  labs(x = "covariate", y="composition") +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25)

g2_gam_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
```

```

    title = element_text(size=8)) +
ggtitle("GAM estimated response by species") +
labs(x = "covariate", y="composition") +
geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25) +
facet_wrap(~ species, ncol = 4)
multiplot(g1_gam_post, g2_gam_post, gsim1, gsim2, cols=2)

```



### S1.3.3 Generate GAM predictions

Like the MVGP model, we generate predictions using the GAM model using a two-stage estimation process.

```

## Note, there might be some initial warnings about ESS shrunk to the
## current position for the first few iterations. This is not a
## major concern as long as the warnings don't continue.
if (file.exists(paste0(save_directory, "sim-bummer-predict-gam.RData"))) {
  load(paste0(save_directory, "sim-bummer-predict-gam.RData"))
} else {
  pred_gam <- predict_compositional_data(
    dat$y_pred,
    dat$X,
    params,
    samples_gam,
    progress_directory,
    "sim-bummer-predict-gam.txt")
  save(pred_gam, file = paste0(save_directory, "sim-bummer-predict-gam.RData"))
}

```

### S1.3.4 Plot GAM predictions

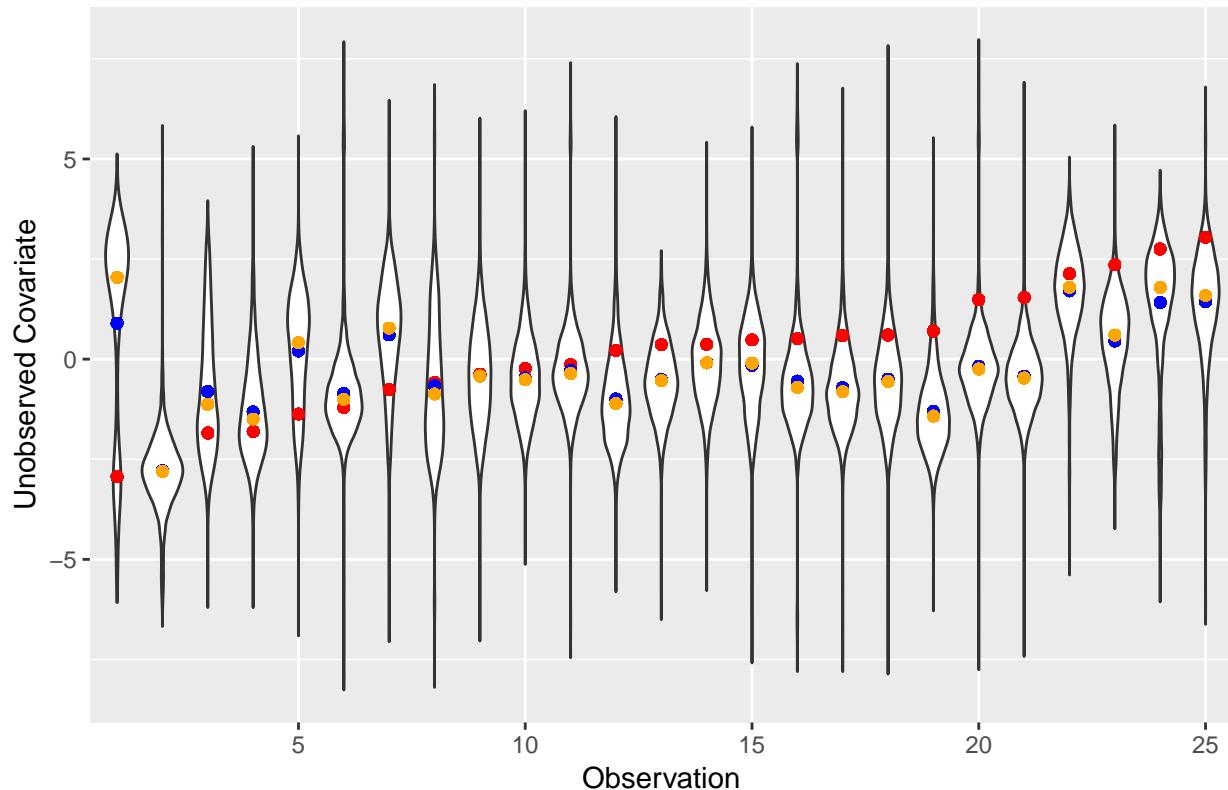
The predictions for 25 randomly chosen missing covariate observations are shown below. The simulated covariate values are shown as red dots and the posterior distribution shown as a violin plot where the width of the violin is proportional to the posterior density. The posterior mean is shown in blue and the posterior median is shown in orange. In general, the posterior distributions are doing a good job of estimating the unobserved covariate. Notice that there is still some multi-modality in the posterior estimates; however, the amount of multi-modality is less.

```
## plot the same 25 observations as the MVGP model
n_samples <- length(pred_gam$X[, 1])

sim_gam_df <- data.frame(
  Covariate = c(pred_gam$X[, idx[plot_idx]]),
  Mean      = rep(apply(pred_gam$X[, idx[plot_idx]], 2, mean), each = n_samples),
  Median    = rep(apply(pred_gam$X[, idx[plot_idx]], 2, median), each = n_samples),
  Observation = factor(rep(1:n_plot, each = n_samples )),
  truth      = rep(dat$X_pred[idx[plot_idx]], each = n_samples))

ggplot(sim_gam_df, aes(Observation, Covariate)) +
  geom_violin(position="identity") +
  geom_point(aes(Observation, truth), color="red") +
  geom_point(aes(Observation, Mean), color="blue") +
  geom_point(aes(Observation, Median), color="orange") +
  scale_x_discrete(breaks=seq(5, n_plot, 5)) +
  labs(x="Observation", y="Unobserved Covariate") +
  ggtitle("GAM prediction for BUMMER simulated data")
```

GAM prediction for BUMMER simulated data



## S1.4 Fitting the BUMMER model

The BayesComposition package has an option for fitting the BUMMER model. We fit the model for 200,000 MCMC iterations, discarding the first 50,000 iterations with adaptive Metropolis-Hastings proposals as burn in and thinning every 150 iterations. The model is run for 4 chains resulting in 4000 posterior samples.

```
## define the model parameters for the GAM model
params <- list(
  n_adapt          = 50000,
  n_mcmc           = 150000,
  n_thin            = 150,
  likelihood        = "dirichlet-multinomial",
  function_type     = "bummer",
  n_chains          = 4,
  n_cores           = 4)

if (file.exists(paste0(save_directory, "sim-bummer-fit-bummer.RData"))) {
  load(paste0(save_directory, "sim-bummer-fit-bummer.RData"))
} else {
  ## potentially long running MCMC code
  out_bummer <- fit_compositional_data(
    y                  = dat$y,
    X                  = dat$X,
    params             = params,
    progress_directory = progress_directory,
    progress_file      = "sim-bummer-fit-bummer.txt")
  ## save the file
  save(out_bummer, file = paste0(save_directory, "sim-bummer-fit-bummer.RData"))
}
```

### S1.4.1 BUMMER convergence diagnostics

After fitting the BUMMER model, the posterior samples are generated and MCMC convergence is assessed using the Gelman-Rubin  $\hat{R}$  statistic. We see the distribution of  $\hat{R}$  statistics is close to 1 with the largest  $\hat{R} = 1.0075429 \leq 1.1$  at an acceptable value (Gelman et al. [2013], pp.287).

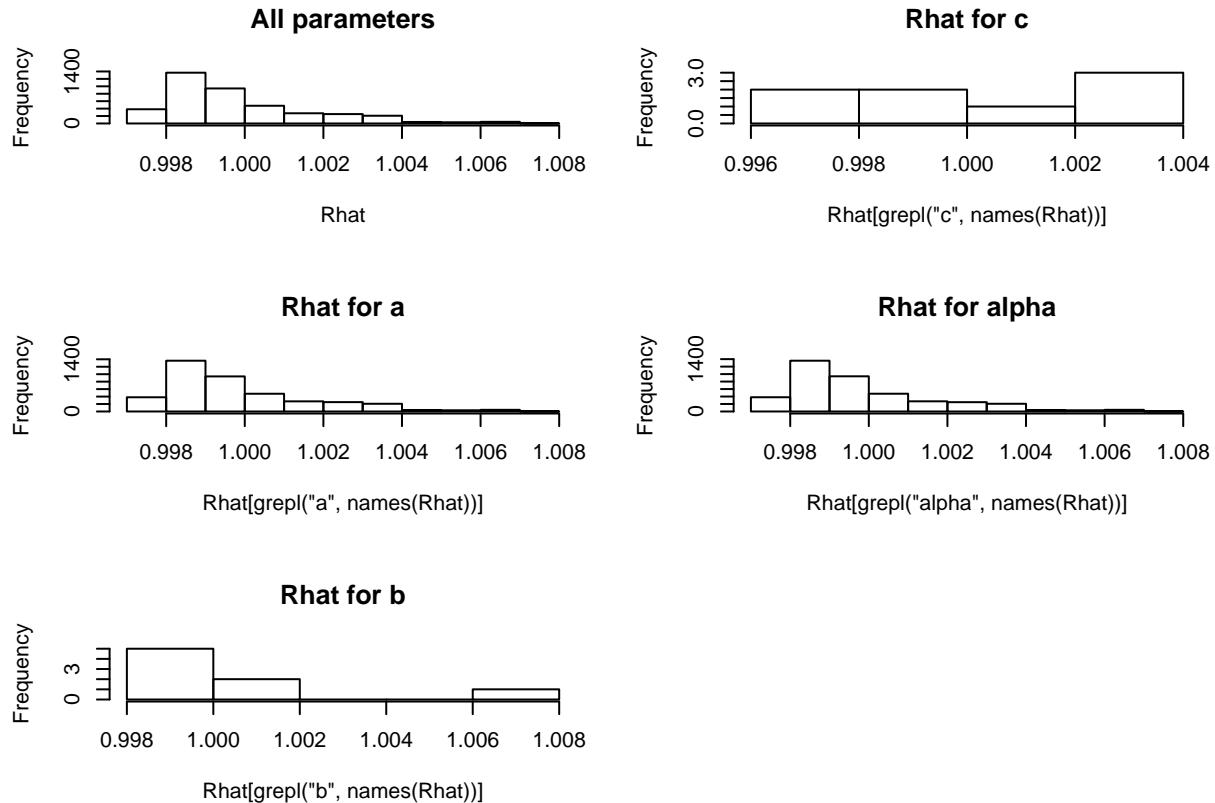
```
## extract posterior samples
samples_bummer <- extract_compositional_samples(out_bummer)
a_post <- samples_bummer$a
b_post <- samples_bummer$b
c_post <- samples_bummer$c
alpha_post <- samples_bummer$alpha

n_samples <- nrow(a_post)

## Calculate Gelman-Rubin convergence statistic
Rhat <- make_gelman_rubin(out_bummer)
layout(matrix(1:6, 3, 2))
hist(Rhat, main="All parameters")
hist(Rhat[grep("a", names(Rhat))], main = "Rhat for a")
hist(Rhat[grep("b", names(Rhat))], main = "Rhat for b")
hist(Rhat[grep("c", names(Rhat))], main = "Rhat for c")
hist(Rhat[grep("alpha", names(Rhat))], main = "Rhat for alpha")
```

```
Rhat[!is.finite(Rhat)] <- NA
max(unlist(na.omit(Rhat)))
```

```
## [1] 1.007543
```



### S1.4.2 BUMMER functional response estimation

We compare how well the BUMMER model reconstructs the simulated functional relationships below. The BUMMER model estimates the simulated functional response well which is not surprising given the data are simulated from the BUMMER model.

```
alpha_post_mean <- apply(alpha_post, c(2, 3), mean)

p_alpha_post_mean <- t(sapply(1:N, function(i) {
  alpha_post_mean[i, ] / sum(alpha_post_mean[i, ])
}))

fitPlotData <- data.frame(species=as.factor(rep(1:d, each=N)),
                           count=c(y_prop),
                           depth=rep(dat$X, times=d),
                           alpha=c(p_alpha_post_mean))

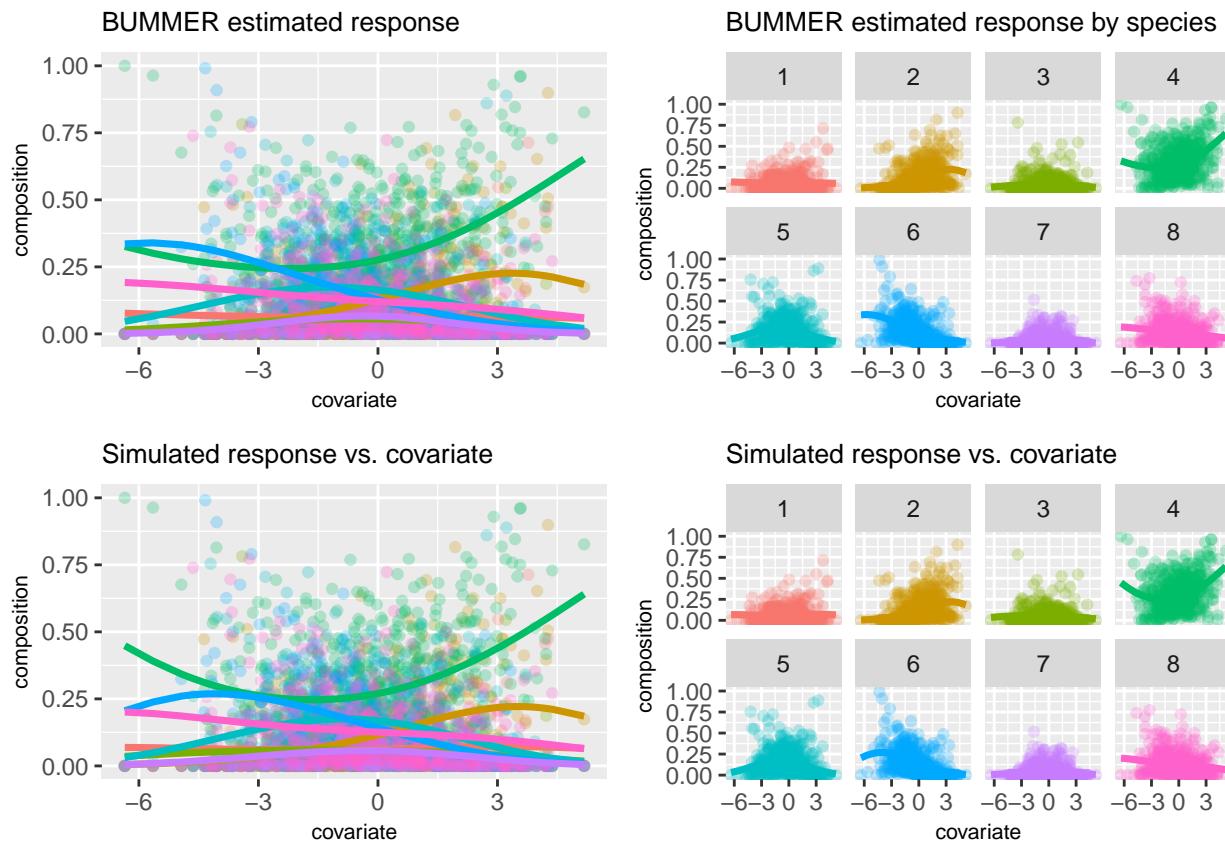
g1_gam_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
        title = element_text(size=8)) +
  ggtitle("BUMMER estimated response") +
  labs(x = "covariate", y="composition") +
```

```

geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25)

g2_gam_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
        title = element_text(size=8)) +
  ggtitle("BUMMER estimated response by species") +
  labs(x = "covariate", y="composition") +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25) +
  facet_wrap(~ species, ncol = 4)
multiplot(g1_gam_post, g2_gam_post, gsim1, gsim2, cols=2)

```



### S1.4.3 Generate BUMMER predictions

We generate predictions using the BUMMER model using a two-stage estimation process.

```

## Note, there might be some initial warnings about ESS shrunk to the
## current position for the first few iterations. This is not a
## major concern as long as the warnings don't continue.
if (file.exists(paste0(save_directory, "sim-bummer-predict-bummer.RData"))) {
  load(paste0(save_directory, "sim-bummer-predict-bummer.RData"))
} else {
  pred_bummer <- predict_compositional_data(
    dat$y_pred,
    dat$X,
    params,

```

```

    samples_bummer,
    progress_directory,
    "sim-bummer-predict-bummer.txt")
save(pred_bummer, file = paste0(save_directory, "sim-bummer-predict-bummer.RData"))
}

```

#### S1.4.4 Plot BUMMER predictions

The predictions for 25 randomly chosen missing covariate observations are shown below. The simulated covariate values are shown as red dots and the posterior distribution shown as a violin plot where the width of the violin is proportional to the posterior density. The posterior mean is shown in blue and the posterior median is shown in orange. In general, the posterior distributions are doing a good job of estimating the unobserved covariate. Notice that there is some multi-modality in the posterior estimates from BUMMER as well.

```

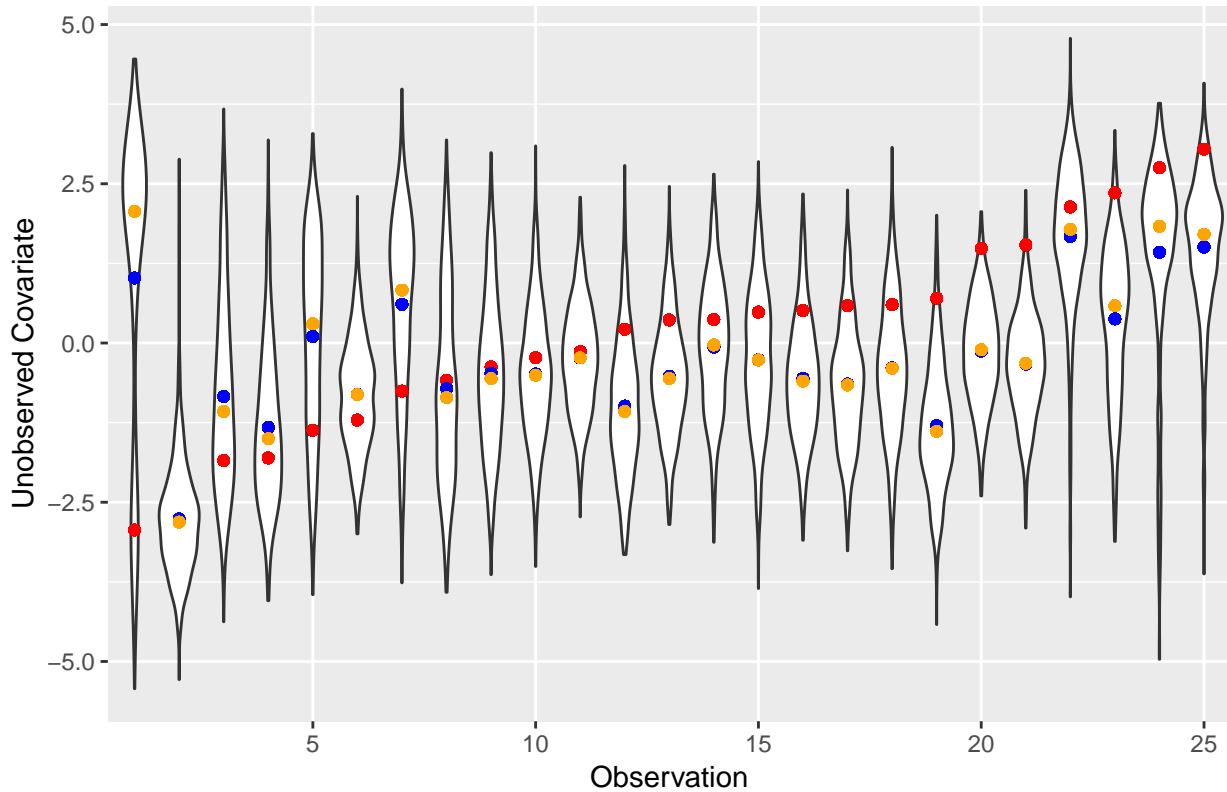
## plot the same 25 observations as the MVGP model
n_samples <- length(pred_bummer$X[, 1])

sim_bummer_df <- data.frame(
  Covariate = c(pred_bummer$X[, idx[plot_idx]]),
  Mean = rep(apply(pred_bummer$X[, idx[plot_idx]], 2, mean), each = n_samples),
  Median = rep(apply(pred_bummer$X[, idx[plot_idx]], 2, median), each = n_samples),
  Observation = factor(rep(1:n_plot, each = n_samples)),
  truth = rep(dat$X_pred[idx[plot_idx]], each = n_samples))

ggplot(sim_bummer_df, aes(Observation, Covariate)) +
  geom_violin(position="identity") +
  geom_point(aes(Observation, truth), color="red") +
  geom_point(aes(Observation, Mean), color="blue") +
  geom_point(aes(Observation, Median), color="orange") +
  scale_x_discrete(breaks=seq(5, n_plot, 5)) +
  labs(x="Observation", y="Unobserved Covariate") +
  ggtitle("BUMMER prediction for BUMMER simulated data")

```

## BUMMER prediction for BUMMER simulated data



### S1.5 Fitting the transfer function models to the simulated data

Below we fit the transfer function models MAT, WA, and MLRC to the simulated data and evaluate the scoring statistics for each model.

```
if (file.exists(paste0(save_directory, "sim-bummer-all-predictions.RData"))) {
  load(paste0(save_directory, "sim-bummer-all-predictions.RData"))
} else {
  X_train <- dat$X
  X_test <- dat$X_pred
  y_train <- dat$y
  y_train_prop <- y_train
  y_test <- dat$y_pred
  y_test_prop <- y_test

  ## convert from counts to proportions
  for (i in 1:N) {
    y_train_prop[i, ] <- y_train_prop[i, ] / sum(y_train_prop[i, ])
  }
  for (i in 1:N_pred) {
    y_test_prop[i, ] <- y_test_prop[i, ] / sum(y_test_prop[i, ])
  }
  colnames(y_train_prop) <- letters[1:d]
  colnames(y_test_prop) <- letters[1:d]

  ##
```

```

## evaluate predictive ability for MVGP
## 

CRPS_MVGP <- makeCRPS(pred$X, X_test, n_samples)
MSPE_MVGP <- (apply(pred$X, 2, mean) - X_test)^2
MAE_MVGP <- abs(apply(pred$X, 2, median) - X_test)
X_025 <- apply(pred$X, 2, quantile, prob = 0.025)
X_975 <- apply(pred$X, 2, quantile, prob = 0.975)
coverage_MVGP <- (X_test >= X_025) & (X_test <= X_975)

##
## evaluate predictive ability for GAM
## 

CRPS_GAM <- makeCRPS(pred_gam$X, X_test, n_samples)
MSPE_GAM <- (apply(pred_gam$X, 2, mean) - X_test)^2
MAE_GAM <- abs(apply(pred_gam$X, 2, median) - X_test)
X_025 <- apply(pred_gam$X, 2, quantile, prob = 0.025)
X_975 <- apply(pred_gam$X, 2, quantile, prob = 0.975)
coverage_GAM <- (X_test >= X_025) & (X_test <= X_975)

##
## evaluate predictive ability for BUMMER
## 

CRPS_BUMMER <- makeCRPS(pred_bummer$X, X_test, n_samples)
MSPE_BUMMER <- (apply(pred_bummer$X, 2, mean) - X_test)^2
MAE_BUMMER <- abs(apply(pred_bummer$X, 2, median) - X_test)
X_025 <- apply(pred_bummer$X, 2, quantile, prob = 0.025)
X_975 <- apply(pred_bummer$X, 2, quantile, prob = 0.975)
coverage_BUMMER <- (X_test >= X_025) & (X_test <= X_975)

##
## evaluate predictive ability for WA
## 

## WA reconstruction - subset to deal with all zero occurrence species
zeros_idx <- which(colSums(y_train_prop) == 0)
if (length(zeros_idx) > 0) {
  modWA <- rioja::WA(y_train_prop[, - zeros_idx], X_train)
  predWA <- predict(modWA, y_test_prop[, - zeros_idx], sse=TRUE, nboot=1000)
} else {
  ## no data to subset
  modWA <- rioja::WA(y_train_prop, X_train)
  predWA <- predict(modWA, y_test_prop, sse=TRUE, nboot=1000)
}

CRPS_WA <- abs(predWA$fit[, 1] - X_test)
MAE_WA <- abs(predWA$fit[, 1] - X_test)
MSPE_WA <- (predWA$fit[, 1] - X_test)^2
coverage_WA <-
  (X_test >= (predWA$fit[, 1] -
    2 * sqrt(predWA$v1.boot[, 1]^2 + predWA$v2.boot[1]^2))) &

```

```

(X_test <= (predWA$fit[, 1] +
             2 * sqrt(predWA$v1.boot[, 1]^2 + predWA$v2.boot[1]^2)))

## 
## evaluate predictive ability for MLRC
## 

## MLRC reconstruction - subset to deal with all zero occurrence species
zeros_idx <- which(colSums(y_train_prop) == 0)
if (length(zeros_idx) > 0) {
  modMLRC <- rioja::MLRC(y_train_prop[, - zeros_idx], X_train)
  predMLRC <- predict(modMLRC, y_test_prop[, - zeros_idx],
                       sse=TRUE, nboot=1000)
} else {
  modMLRC <- rioja::MLRC(y_train_prop, X_train)
  predMLRC <- predict(modMLRC, y_test_prop, sse=TRUE, nboot=1000)
}
CRPS_MLRC <- abs(predMLRC$fit[, 1] - X_test)
MSPE_MLRC <- (predMLRC$fit[, 1] - X_test)^2
MAE_MLRC <- abs(predMLRC$fit[, 1] - X_test)
coverage_MLRC <-
  (X_test >= (predMLRC$fit[, 1] -
               2 * sqrt(predMLRC$v1.boot[, 1]^2 + predMLRC$v2.boot[1]^2))) &
  (X_test <= (predMLRC$fit[, 1] +
               2 * sqrt(predMLRC$v1.boot[, 1]^2 + predMLRC$v2.boot[1]^2)))

## 
## evaluate predictive ability for MAT
## 

## Modern analogue technique
modMAT <- rioja::MAT(as.data.frame(y_train_prop), X_train, lean = FALSE)
predMAT <- predict(modMAT, as.data.frame(y_test_prop), sse=TRUE, n.boot=1000)
CRPS_MAT <- abs(predMAT$fit[, 1] - X_test)
MSPE_MAT <- (predMAT$fit[, 1] - X_test)^2
MAE_MAT <- abs(predMAT$fit[, 1] - X_test)
coverage_MAT <-
  (X_test >= (predMAT$fit.boot[, 2] -
               2 * sqrt(predMAT$v1.boot[, 2]^2 + predMAT$v2.boot[2]))) &
  (X_test <= (predMAT$fit.boot[, 2] +
               2 * sqrt(predMAT$v1.boot[, 2]^2 + predMAT$v2.boot[2]))))

save(
  CRPS_MVGP, MSPE_MVGP, MAE_MVGP, coverage_MVGP,
  CRPS_GAM, MSPE_GAM, MAE_GAM, coverage_GAM,
  CRPS_BUMMER, MSPE_BUMMER, MAE_BUMMER, coverage_BUMMER,
  modWA, predWA, CRPS_WA, MAE_WA, MSPE_WA, coverage_WA,
  modMLRC, predMLRC, CRPS_MLRC, MSPE_MLRC, MAE_MLRC, coverage_MLRC,
  modMAT, predMAT, CRPS_MAT, MSPE_MAT, MAE_MAT, coverage_MAT,
  file = paste0(save_directory, "sim-bummer-all-predictions.RData"))
}

```

### S1.5.1 Prediction results

Finally, we generate the results from the simulation study. The results below show that for data simulated from the BUMMER model, the BUMMER model generates the best predictions across all metrics; however the MVGP and GAM model predictions perform nearly as well. The transfer function methods perform worse than the probabilistic methods across the board.

```
CRPS_out <- cbind(CRPS_MVGP, CRPS_GAM, CRPS_BUMMER,
                     CRPS_WA, CRPS_MAT, CRPS_MLRC)
MSPE_out <- cbind(MSPE_MVGP, MSPE_GAM, MSPE_BUMMER,
                     MSPE_WA, MSPE_MAT, MSPE_MLRC)
MAE_out <- cbind(MAE_MVGP, MAE_GAM, MAE_BUMMER,
                     MAE_WA, MAE_MAT, MAE_MLRC)
coverage_out <- cbind(coverage_MVGP, coverage_GAM, coverage_BUMMER,
                      coverage_WA, coverage_MAT, coverage_MLRC)
colnames(CRPS_out) <- c("MVGP", "GAM", "BUMMER", "WA", "MAT", "MLRC")
colnames(MAE_out) <- c("MVGP", "GAM", "BUMMER", "WA", "MAT", "MLRC")
colnames(MSPE_out) <- c("MVGP", "GAM", "BUMMER", "WA", "MAT", "MLRC")
colnames(coverage_out) <- c("MVGP", "GAM", "BUMMER", "WA", "MAT", "MLRC")

CRPS <- data.frame(t(apply(CRPS_out, 2, mean)))
MSPE <- data.frame(t(apply(MSPE_out, 2, mean)))
MAE <- data.frame(t(apply(MAE_out, 2, mean)))
coverage <- data.frame(100/(N_pred)*t(apply(coverage_out, 2, sum)))

sim_results <- t(rbind(CRPS, MSPE, MAE, coverage))
colnames(sim_results) <- c("CRPS", "MSPE", "MAE", "95% CI coverage")
## print the results for table 1
# print(xtable(sim_results, digits=4), file="~/mvgp/results/sim-bummer.tex",
#       floating=FALSE)
kable(sim_results)
```

	CRPS	MSPE	MAE	95% CI coverage
MVGP	0.6502221	1.400818	0.9104588	95.0
GAM	0.6456110	1.368344	0.9121387	96.0
BUMMER	0.6397339	1.348020	0.9153544	93.5
WA	1.1142604	1.881387	1.1142604	98.0
MAT	1.1057374	1.999319	1.1057374	98.5
MLRC	1.6330897	4.329101	1.6330897	100.0

## References

- Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2013.
- Iain Murray, Ryan Prescott Adams, and David J. C. MacKay. Elliptical slice sampling. In *AISTATS*, volume 13, pages 541–548, 2010.
- Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009.