Appendix S3: Analysis of testate amoebae data

S3.1 Fitting the models to the testate amoebae dataset

This appendix is broken into five sections. In the first section, we load and pre-process the testate amoebae data. In the next three sections, we fit the probabilistic models of MVGP, GAM, and BUMMER to the testate amoebae dataset with 25 observations held-out to compare the posterior predictive distributions to the held-out observations. The final section provides code for replication of the cross-validation study for the testate amoebae data set.

To evaluate the performance of the proposed inverse prediction framework, we fit the proposed models to the testate amoebae dataset consisting of 356 observations of a composition of 24 species along with measurements of water table depth in cm. First, we load the necessary `R` packages and setting up the file directories for saving model output and progress files.

```
library(BayesComposition)
library(here)
library(ggplot2)
library(GGally)
library(kableExtra)
library(reshape2)
library(snowfall)
library(parallel)
library(rlecuyer)
library(xtable)
library(knitr)

## 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/"

## Number of observations to randomly hold out in first stage fit
N_pred <- 25
```

S3.2 Load testate amoebae data

Next, we load the testate amoebae data. The testate amoebae data consists of 356 observations of 24 different taxonomic units (we simplify the language and call them "species"). The composition is a vector of counts and is displayed below as a relative proportion with each species having a distinct color. For fitting of the models, we center and scale the covariate and plot the testate amoebae dataset.

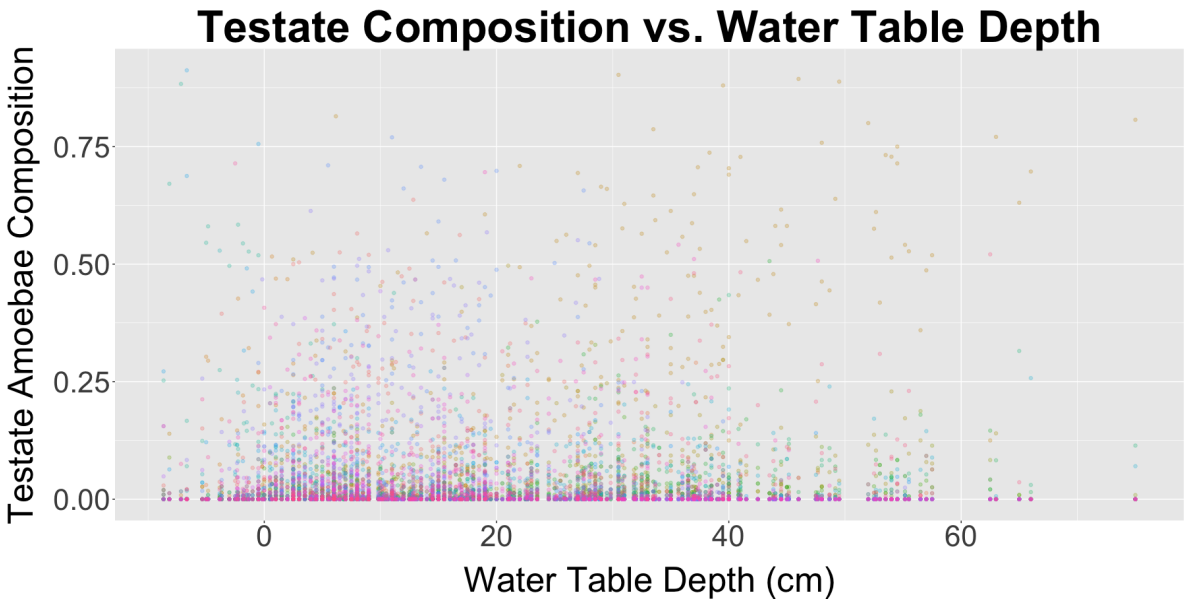
```
raw_data <- read.csv(
  file = here::here("data",
                    "North American Raw Testate - Paleon 2017-Sheet1.csv"),
  skip=6)

## subset to Booth 2008
raw_data <- raw_data[1:378, ]
```

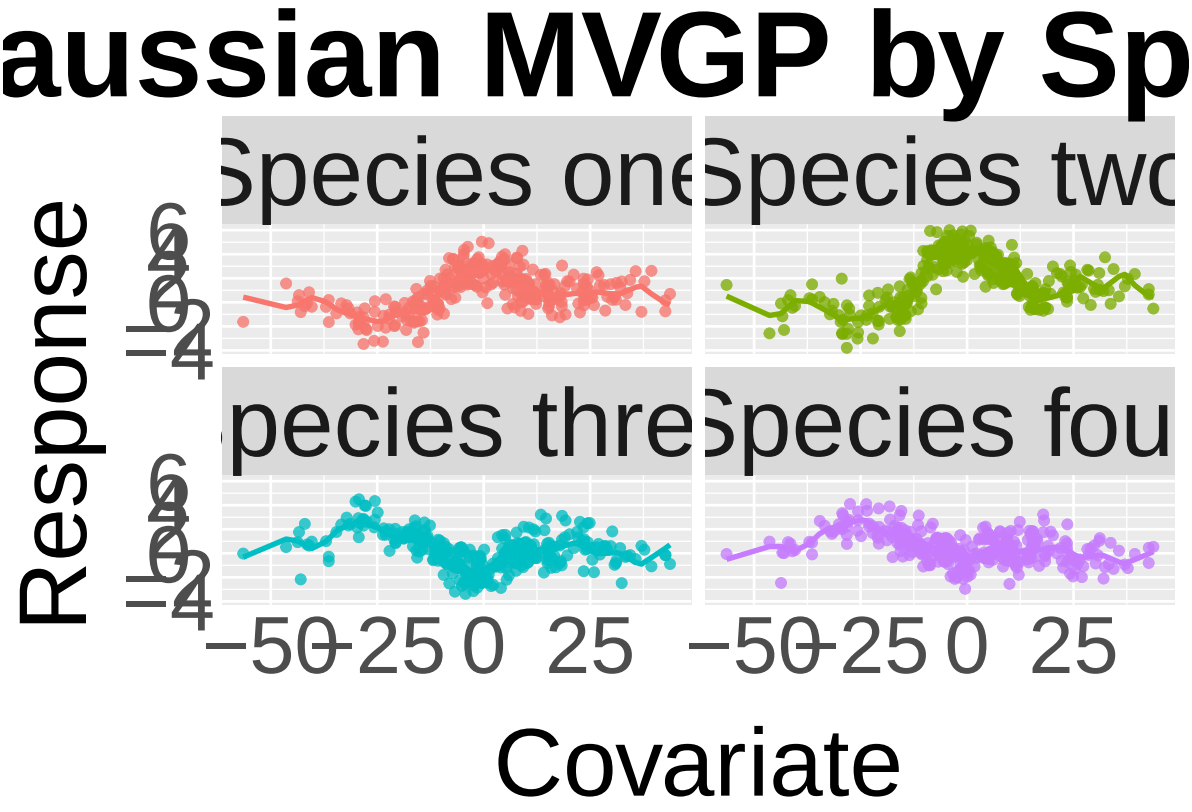
```
y <- raw_data[, 12:85]
X <- raw_data$WTD..cm.

## Subset to Booth 2008 data
N <- 356
N_obs <- 356
y <- y[1:356, ]
X <- X[1:356]

## join species together according to expert opinion
y <- join_testate_booth(y)

## remove species with zero count
no_obs <- which(base::colSums(y) == 0)

## down to 47 species
y <- y[, -no_obs]

## Subset rare species
sum(y)
```

```
## [1] 65981
```

```
y <- y[, base::colSums(y) > 500]

N <- dim(y)[1]
d <- dim(y)[2]

y <- as.matrix(y)

## center the covariates for algorithm stability
mean_X <- mean(X)
sd_X <- sd(X)
X <- (X - mean_X) / sd_X

## transform the data to percentages for use in transfer function models
y_prop <- y
for (i in 1:N) {
  y_prop[i, ] <- y_prop[i, ] / sum(y_prop[i, ])
}

testatePlotData <- data.frame(species=as.factor(rep(colnames(y), each=N)),
                              Count=c(as.matrix(y_prop)),
                              Wetness=rep(X * sd_X + mean_X, times=dim(y)[2]))

png(file = paste0(save_directory, "testate-plot.png"), width = 18, height = 9,
    units = "in", res = 100)
ggplot(testatePlotData, aes(x=Wetness, y=Count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme() +
  ggtitle("Testate Composition vs. Water Table Depth") +
  labs(x="Water Table Depth (cm)", y="Testate Amoebae Composition") +
  theme(legend.position="none",
        plot.title=element_text(size=48, face="bold", hjust=0.5),
```

```
        axis.text.x = element_text(size = 32),
        axis.text.y = element_text(size = 32),
        axis.title.x = element_text(size = 38,
                                    margin = margin(t = 20, r = 0, b = 0, l = 0)),
        axis.title.y = element_text(size = 38,
                                    margin = margin(t = 0, r = 20, b = 0, l = 0)))
invisible(dev.off())
include_graphics(paste0(save_directory, "testate-plot.png"))
```





S3.3 Fitting the MVGP model to the testate amoebae data

For fitting the MVGP model, we define a set of knots for the predictive process approximation. We chose 30 evenly spaced knots that spanned ±1.25 standard deviations beyond the range of the observed covariate values to prevent any edge effects for predicted values near the extent of the observed values. We fit the MVGP model to the testate amoebae dataset using 200,000 MCMC iterations, discarding the first 50,000 iterations as burn-in with adaptive proposals [Roberts and Rosenthal, 2009]. The remaining 150,000 MCMC iterations are generated with fixed proposal distributions and thinned every 150 iterations to reduce file size for 4 parallel chains resulting in 4,000 posterior samples. We used an exponential covariance structure for the Gaussian processes while estimating a multiplicative functional correlation. We did not include an additional additive random effect $\boldsymbol{\varepsilon}$ to model overdispersion, although this could be included for other datasets. To improve the estimation of the correlation of functional responses, we set a prior on the Cholesky factor of the functional correlation $\boldsymbol{R} \sim \text{LKJ}(8)$ which applies shrinkage of the functional correlation parameters towards 0 [Lewandowski et al., 2009]. The MVGP model took 8.8 hours running 4 MCMC chains in parallel on a 2017 iMac with 4.2GHz processor.

```
## define the knots
n_knots <- 30
X_knots <- seq(min(X, na.rm=TRUE)-1.25*sd(X, na.rm=TRUE),
               max(X, na.rm=TRUE)+1.25*sd(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,
  ## prior on covariance to improve estimation
  eta                       = 8,
  additive_correlation      = FALSE,
  n_chains                  = 4,
  n_cores                   = 4,
  n_knots                   = n_knots,
  X_knots                   = X_knots)

if (file.exists(paste0(save_directory, "fit-mvgp-testate.RData"))) {
  ## Load MCMC run
  load(paste0(save_directory, "fit-mvgp-testate.RData"))
} else {

  ##
  ## Long running MCMC
  ##

  ## time the MCMC
  start <- Sys.time()

  set.seed(404)
  ## set the sample idx so the held-out observations are the same
  sample_idx <- sample(1:N, N_pred)
```

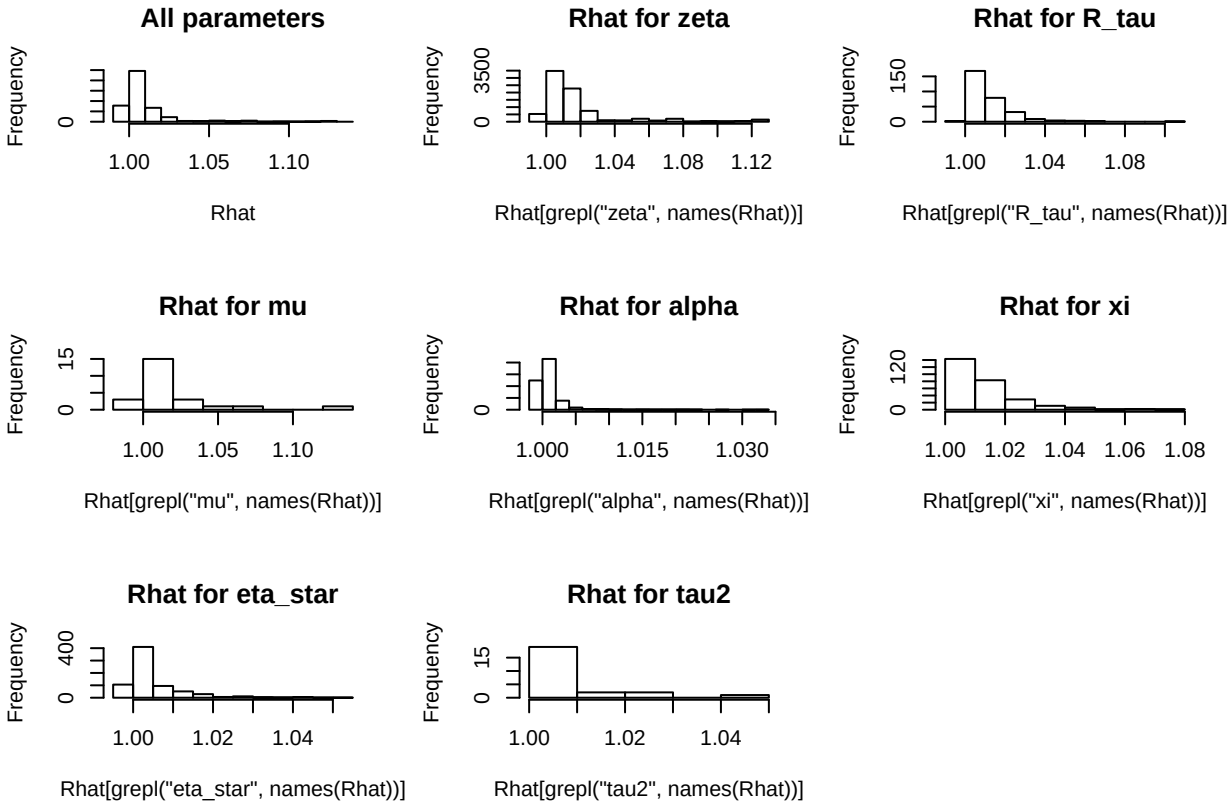
```
  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                 = y[ - sample_idx, ],
    X                 = X[ - sample_idx],
    params            = params,
    progress_directory = progress_directory,
    progress_file     = "fit-mvgp-testate.txt")

  ## time the MCMC
  finish <- Sys.time()
  finish - start
  save(out, start, finish,
       sample_idx, file=paste0(save_directory, "fit-mvgp-testate.RData"))
}
```

S3.3.1 MVGP convergence diagnostics

After fitting the MVGP 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 $\hat{R} = 1.1320552 \lessapprox 1.1$, close to an an acceptable value that would be reached by running more iterations (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$zeta
alpha_post <- samples$alpha
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[grepl("mu", names(Rhat))], main = "Rhat for mu")
hist(Rhat[grepl("eta_star", names(Rhat))], main = "Rhat for eta_star")
hist(Rhat[grepl("zeta", names(Rhat))], main = "Rhat for zeta")
hist(Rhat[grepl("alpha", names(Rhat))], main = "Rhat for alpha")
hist(Rhat[grepl("tau2", names(Rhat))], main = "Rhat for tau2")
hist(Rhat[grepl("R_tau", names(Rhat))], main = "Rhat for R_tau")
hist(Rhat[grepl("xi", names(Rhat))], main = "Rhat for xi")
Rhat[!is.finite(Rhat)] <- NA
max(unlist(na.omit(Rhat)))
```

```
## [1] 1.132055
```



S3.3.2 Generate MVGP predictions

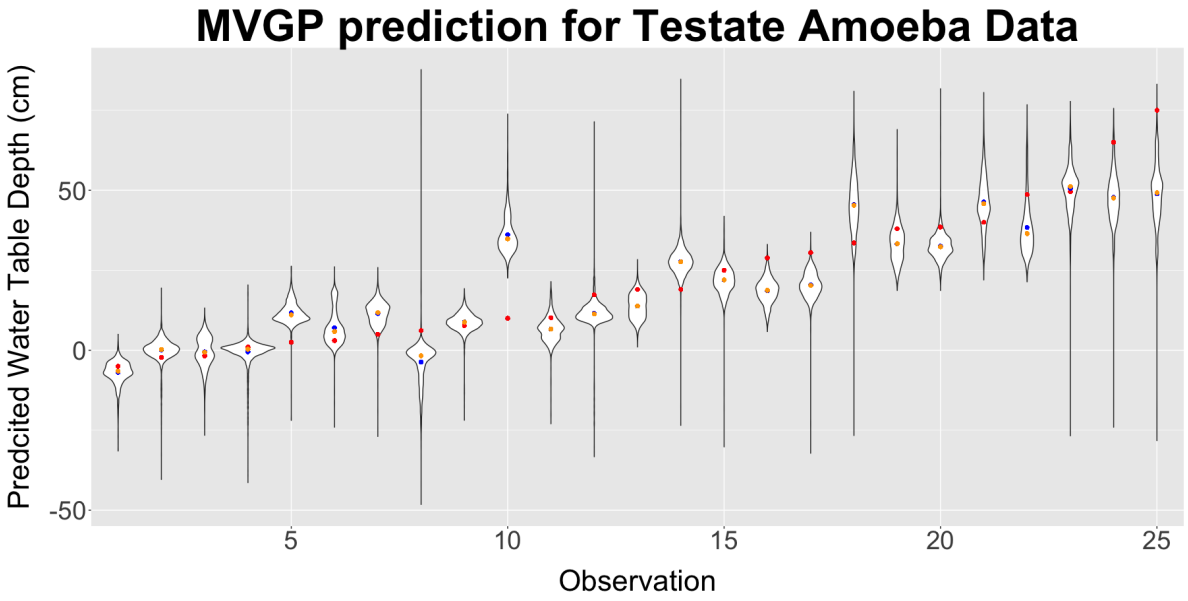
Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the held-out water table depths using the posterior samples estimated from the calibration model. These estimates can be generated either jointly with the calibration model or estimated using a two-stage approach 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, "predict-mvgp-testate.RData"))) {
  load(paste0(save_directory, "predict-mvgp-testate.RData"))
} else {
  pred <- predict_compositional_data(
    y[sample_idx, ],
    X[ - sample_idx],
    params,
    samples,
    progress_directory,
    "predict-mvgp-testate.txt")
  save(pred, file = paste0(save_directory, "predict-mvgp-testate.RData"))
}
```

S3.3.3 Plot MVGP predictions

The predictions for 25 randomly chosen hold-out water table depth observations are shown below. The held-out average water table depth 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. d

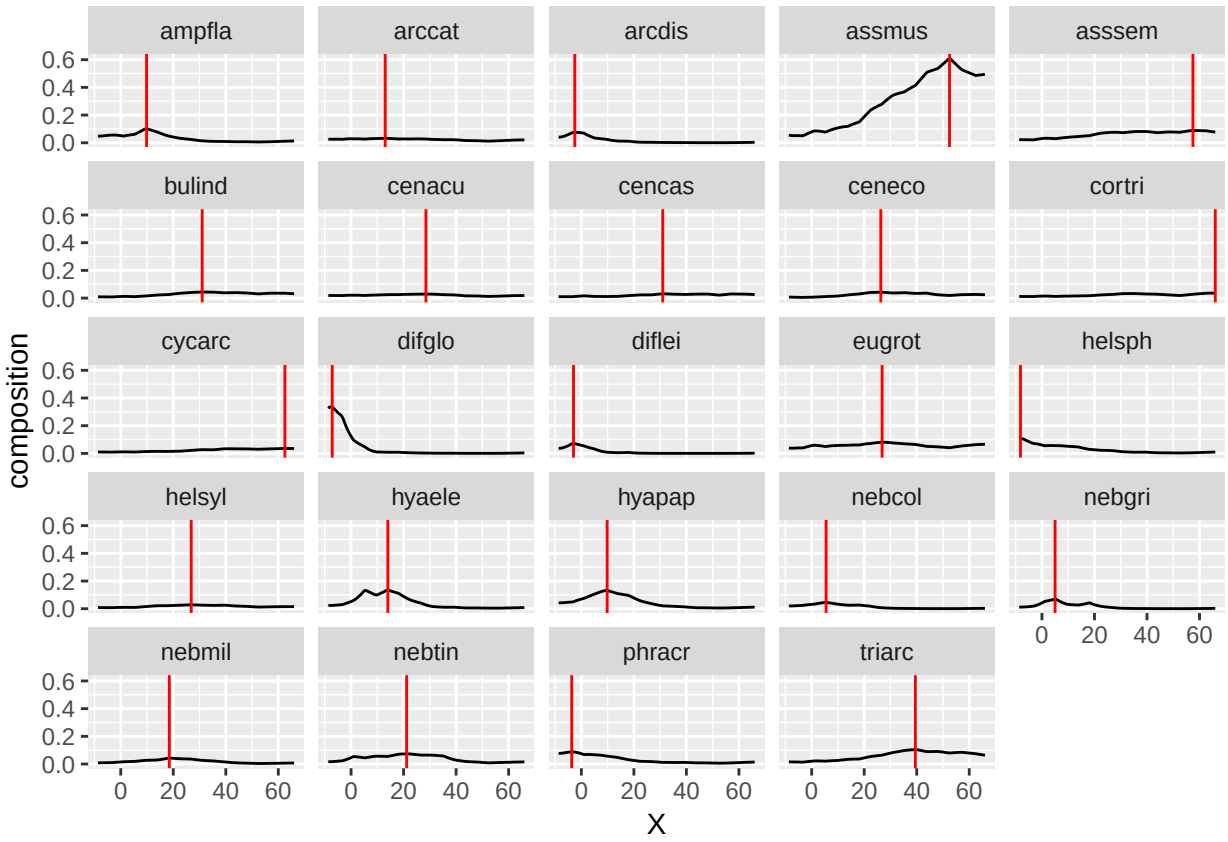
```
## sorted to increasing values for ease of display
idx <- order(X[sample_idx])
## randomly choose 25 observations to display

n_plot <- 25
n_samples <- length(pred$X[, 1])

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

png(file = paste0(save_directory, "testate-predictions.png"), width = 18,
    height = 9, units = "in", res = 100)

ggplot(sim_df, aes(Observation, Covariate)) +
  geom_violin(position="identity", width = 1.25) +
  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="Predcited Water Table Depth (cm)") +
  ggtitle("MVGP prediction for Testate Amoeba Data") +
  theme(plot.title=element_text(size=48, face="bold", hjust=0.5),
        axis.text.x = element_text(size = 28),
        axis.text.y = element_text(size = 28),
        axis.title.x = element_text(
          size   = 32,
          margin = margin(t = 20, r = 0, b = 0, l = 0)
        ),
        axis.title.y = element_text(
          size   = 32,
          margin = margin(t = 0, r = 20, b = 0, l = 0)
        )
  )

invisible(dev.off())
include_graphics(paste0(save_directory, "testate-predictions.png"))
```



S3.3.4 MVGP maximal functional response

We predict the functional responses for each species and order the species by their maximal response for easier presentation of the estimated correlation matrix $\mathbf{\Omega}$. The maximal fitted response for each species is shown by the red vertical line below with the black line representing the latent functional response of the species to climate relative to the remaining composition.

```
## Order species by predicted maximal response
alpha_post_mean <- apply(alpha_post, c(2, 3), mean)
p_alpha_mean <- t(sapply(1:dim(alpha_post_mean)[1],
                         function(i) alpha_post_mean[i, ] / sum(alpha_post_mean[i, ])))
species_maxima <- rep(0, d)
for(i in 1:d) {
  ## order on alpha_post
  ## choose the first peak if there is a tie (arbitrary rule)
  ## ties typically are due to multiple observations having the same X values
  species_maxima[i] <- X[-sample_idx][which(p_alpha_mean[, i] ==
                                            max(p_alpha_mean[, i]))][1]
}

dat <- data.frame(
  species       = rep(colnames(y), each = N - length(sample_idx)),
  maxima        = rep(species_maxima * sd_X + mean_X, each = N - length(sample_idx)),
  X             = rep(X[-sample_idx] * sd_X + mean_X, times = d),
  composition   = c(p_alpha_mean)
)

ggplot(data = dat, aes(x = X, y=composition)) +
  geom_line() +
  facet_wrap(~ species) +
  geom_vline(aes(xintercept = maxima), color = "red")
```



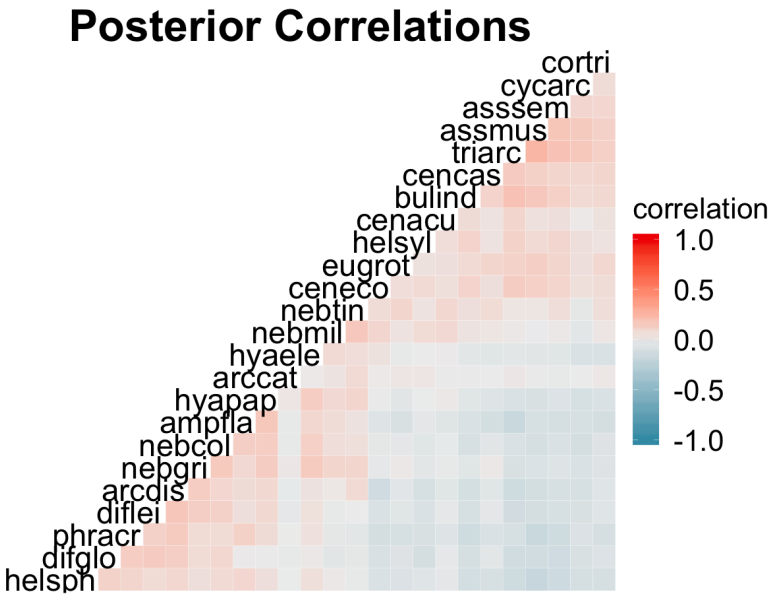

```
order_species <- order(species_maxima, decreasing = FALSE)
```

S3.3.5 Plot MVGP estimated functional correlations

Using the ordering of the species, we plot the estimated correlations in functional response. The posterior mean estimates of the functional correlations can be used to learn about the similarities/differences in response of the different taxonomic units. These functional correlations provide a data-driven method for generating testate functional types for use in future analyses.

```
Omega_post_mean <- matrix(0, d, d)
for (i in 1:n_samples) {
  Omega_post_mean <- Omega_post_mean +
    1/n_samples * t(R_post[i, , ]) %*% R_post[i, , ]
}
colnames(Omega_post_mean) <- colnames(y)

png(file = paste0(save_directory, "testate-correlations.png"), width = 18,
    height = 9, units = "in", res = 100)
ggcorr(data=NULL, cor_matrix=Omega_post_mean[order_species, order_species],
       name="correlation", hjust=0.9, layout.exp=4, size=12) +
  ggtitle("Posterior Correlations") +
  theme(plot.title=element_text(size=48, face="bold", hjust=0.5),
        legend.text=element_text(size=32),
        legend.title=element_text(size=32)) +
  guides(fill=guide_colorbar(barwidth=2, barheight = 16))
invisible(dev.off())
```

```
include_graphics(paste0(save_directory, "testate-correlations.png"))
```



S3.3.6 Plot MVGP estimated functional responses

We also plot the estimated mean functional response for each species with associated Bayesian credible intervals (50% central credible interval is darkly shaded, the 95% central credible interval is lightly shaded).

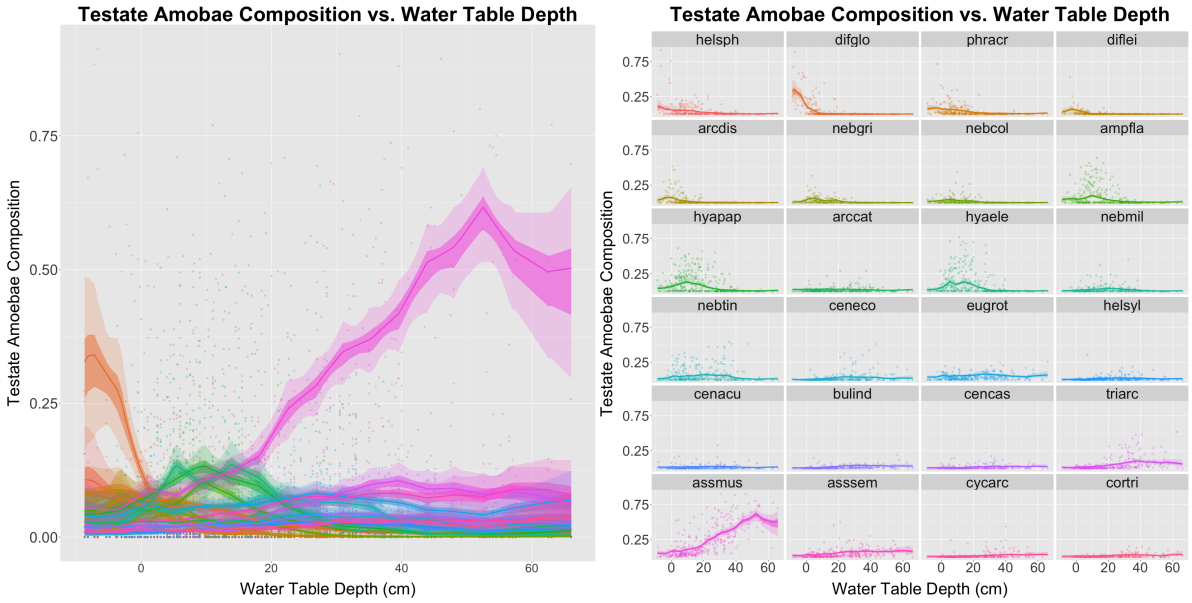
```
alpha_post_mean <- apply(alpha_post, c(2, 3), mean)
alpha_post_sum_to_one <- alpha_post
for (k in 1:n_samples) {
  for (i in 1:(N-length(sample_idx))) {
    alpha_post_sum_to_one[k, i, ] <- alpha_post_sum_to_one[k, i, ] /
      sum(alpha_post_sum_to_one[k, i, ])
  }
}
alpha_post_lower_50 <- apply(alpha_post_sum_to_one, c(2, 3), quantile, prob=0.25)
alpha_post_upper_50 <- apply(alpha_post_sum_to_one, c(2, 3), quantile, prob=0.75)
alpha_post_lower_95 <- apply(alpha_post_sum_to_one, c(2, 3), quantile, prob=0.025)
alpha_post_upper_95 <- apply(alpha_post_sum_to_one, c(2, 3), quantile, prob=0.975)

zeta_post_mean <- apply(zeta_post, c(2, 3), mean)
mu_post_mean <- apply(mu_post, 2, mean)
p_alpha <- matrix(0, N-N_pred, d)
for (i in 1:(N-N_pred)) {
  p_alpha[i, ] <- exp(mu_post_mean + zeta_post_mean[i, ]) / sum(exp(mu_post_mean + zeta_post_mean[i, ]))
}
y_prop <- y
for (i in 1:N) {
  y_prop[i, ] <- y[i, ] / sum(y[i, ])
}

fitPlotData <- data.frame(
  species        = factor(rep(colnames(y), each=N-N_pred),
                          levels=colnames(y)[order_species]),
```

```
  count          = c(y_prop[-sample_idx, ]),
  depth          = rep(X[-sample_idx] * sd_X + mean_X, times=d),
  alpha          = c(p_alpha),
  alpha_lower_50 = c(alpha_post_lower_50),
  alpha_upper_50 = c(alpha_post_upper_50),
  alpha_lower_95 = c(alpha_post_lower_95),
  alpha_upper_95 = c(alpha_post_upper_95))

g1_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  geom_ribbon(aes(ymin=alpha_lower_50, ymax=alpha_upper_50, fill=species, group=species),
              linetype=0, alpha=0.5) +
  geom_ribbon(aes(ymin=alpha_lower_95, ymax=alpha_upper_95, fill=species, group=species),
              linetype=0, alpha=0.2) +
  ggtitle("Testate Amobae Composition vs. Water Table Depth") +
  theme(legend.position="none",
        plot.title=element_text(size=44, face="bold", hjust=0.5),
        axis.text.x = element_text(size = 26),
        strip.text.x = element_text(size = 30),
        axis.text.y = element_text(size = 32),
        axis.title.x = element_text(
          size   = 36,
          margin = margin(t = 20, r = 0, b = 0, l = 0)
        ),
        axis.title.y = element_text(
          size   = 36,
          margin = margin(t = 0, r = 20, b = 0, l = 0)
        )
  ) +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25) +
  labs(x="Water Table Depth (cm)", y="Testate Amoebae Composition")

g2_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  geom_ribbon(aes(ymin=alpha_lower_50, ymax=alpha_upper_50, fill=species, group=species),
              linetype=0, alpha=0.5) +
  geom_ribbon(aes(ymin=alpha_lower_95, ymax=alpha_upper_95, fill=species, group=species),
              linetype=0, alpha=0.2) +
  ggtitle("Testate Amobae Composition vs. Water Table Depth") +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25) +
  facet_wrap( ~ species, ncol = 4) +
  labs(x="Water Table Depth (cm)", y="Testate Amoebae Composition") +
  theme(legend.position="none",
        plot.title=element_text(size=44, face="bold", hjust=0.5,
                  margin = margin(t = 0, r = 0, b = 20, l = 0)),
        axis.text.x = element_text(size = 30),
        strip.text.x = element_text(size = 32),
        axis.text.y = element_text(size = 30),
        axis.title.x = element_text(
          size   = 36,
          margin = margin(t = 20, r = 0, b = 0, l = 0)
        ),
        axis.title.y = element_text(
```

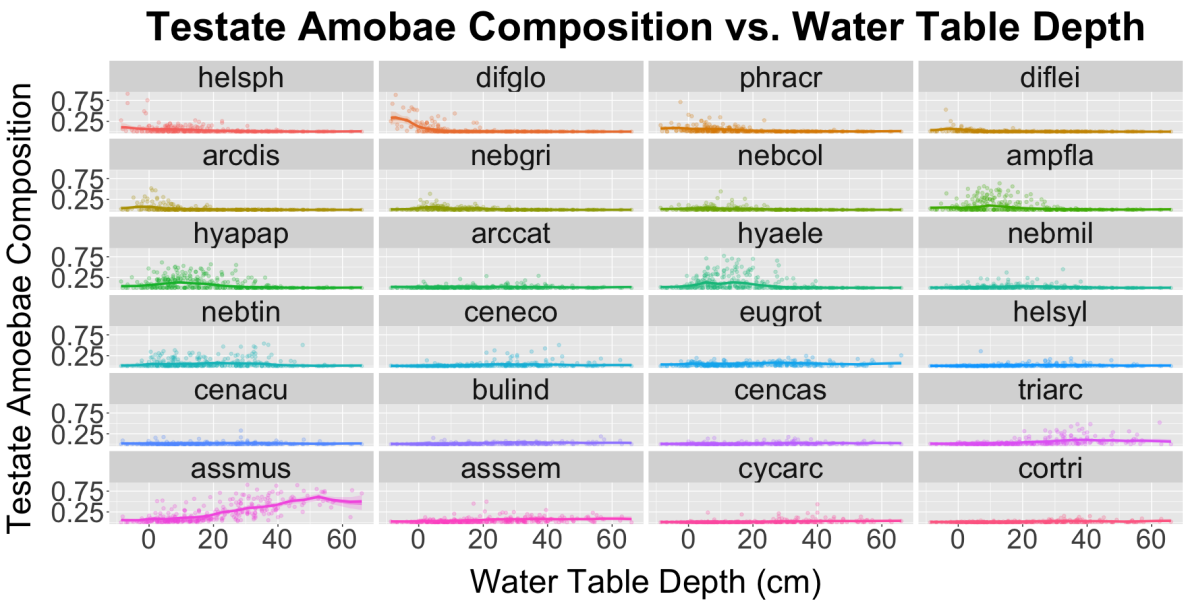
```
          size   = 36,
          margin = margin(t = 0, r = 20, b = 0, l = 0)
        )
  ) +
  scale_y_continuous(breaks=c(0.25, 0.75)) +
  scale_x_continuous(breaks = c(0, 20, 40, 60))

png(file = paste0(save_directory, "testate-fit.png"), width = 18*2,
    height = 9*2, units = "in", res = 100)
multiplot(g1_post, g2_post, cols=2)
```

```
## Loading required package: grid
```

```
invisible(dev.off())

include_graphics(paste0(save_directory, "testate-fit.png"))
```




```
png(file = paste0(save_directory, "testate-fit2.png"), width = 18,
    height = 9, units = "in", res = 100)
g2_post
invisible(dev.off())

include_graphics(paste0(save_directory, "testate-fit2.png"))
```



S3.4 BUMMER model fit to the testate amoebae data

The BUMMER model is included in the `BayesComposition` and we fit the model here. We fit the BUMMER model for 200,000 MCMC iterations, discarding the first 500,000 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
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, "fit-bummer-testate.RData"))) {
  ## Load MCMC run
  load(paste0(save_directory, "fit-bummer-testate.RData"))
} else {
  ##
  ## Long running MCMC
  ##

  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                  = y[ - sample_idx, ],
    X                  = X[ - sample_idx],
    params             = params,
    progress_directory = progress_directory,
    progress_file      = "fit-bummer-testate.txt")
```

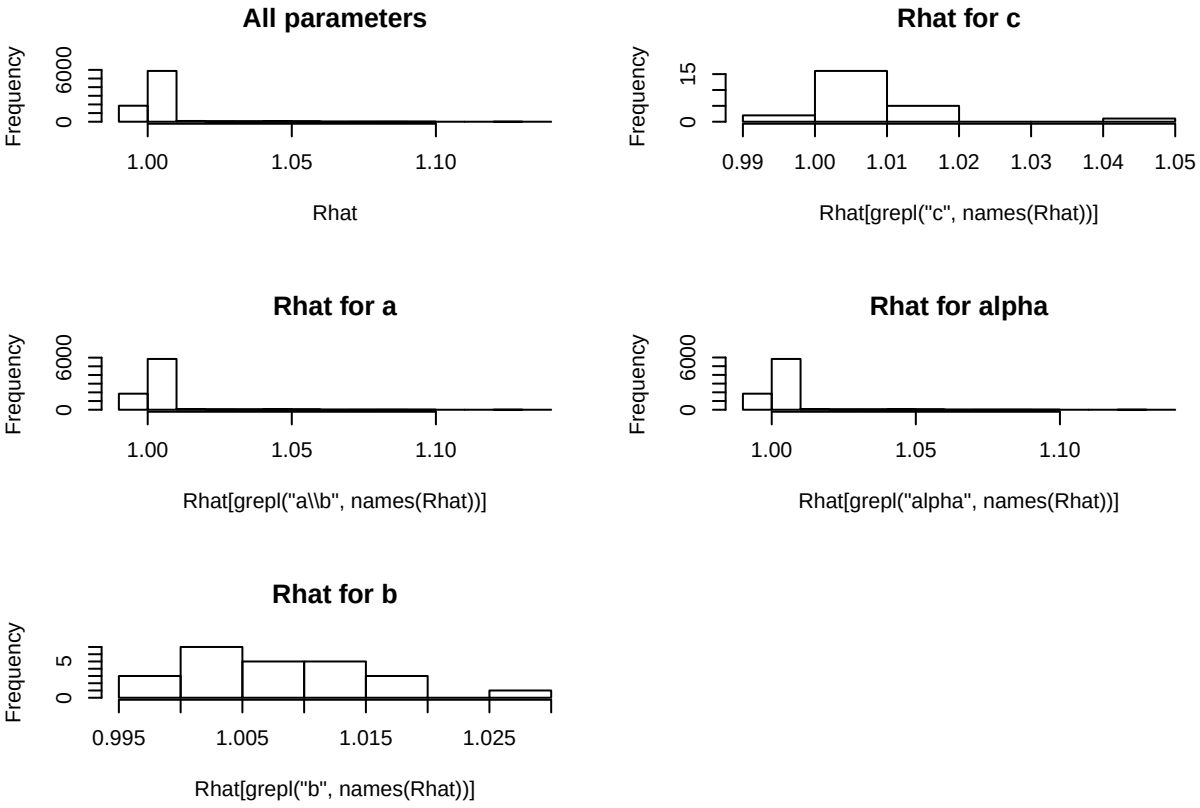
```
  save(out, sample_idx, file = paste0(save_directory, "fit-bummer-testate.RData"))
}
```

S3.4.1 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 $\hat{R} = 1.1363646 \lessapprox 1.1$ at an acceptable value (Gelman et al. [2013], pp.287).

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

n_samples <- nrow(a_post)

## Calculate Gelman-Rubin convergence statistic
Rhat <- make_gelman_rubin(out)
layout(matrix(1:6, 3, 2))
hist(Rhat, main="All parameters")
hist(Rhat[grepl("a\\b", names(Rhat))], main = "Rhat for a")
hist(Rhat[grepl("b", names(Rhat))], main = "Rhat for b")
hist(Rhat[grepl("c", names(Rhat))], main = "Rhat for c")
hist(Rhat[grepl("alpha", names(Rhat))], main = "Rhat for alpha")
Rhat[!is.finite(Rhat)] <- NA
max(unlist(na.omit(Rhat)))
```

```
## [1] 1.136365
```



S3.4.2 Generate BUMMER predictions

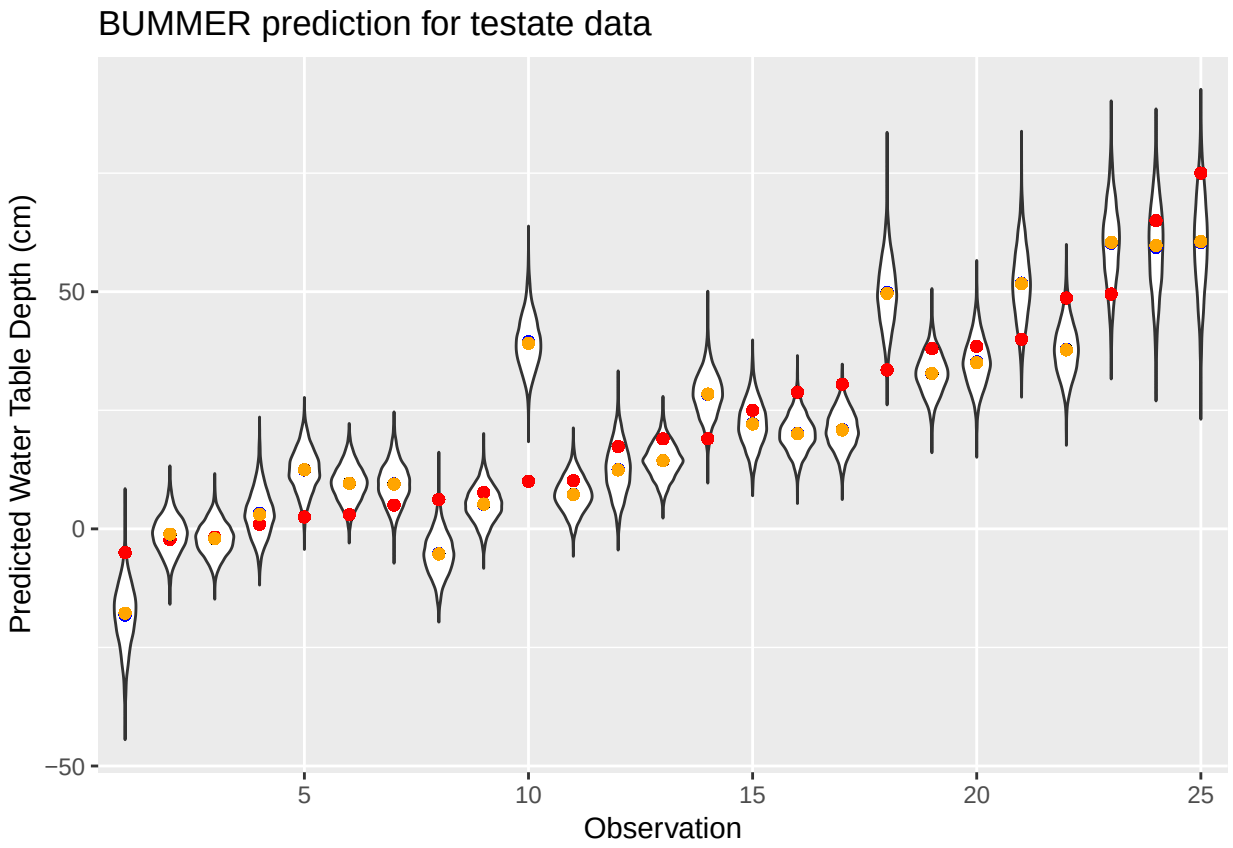
Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved covariates using the posterior samples estimated from the calibration model. These estimates can be generated either jointly with the calibration model or estimated using a two-stage approach 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. There might be some
##    errors because the BUMMER model produces Dirichlet-multinomial parameters
##    alpha that are very close to 0
if (file.exists(paste0(save_directory, "predict-bummer-testate.RData"))) {
  load(paste0(save_directory, "predict-bummer-testate.RData"))
} else {
  pred <- predict_compositional_data(
    y[sample_idx, ],
    X[ - sample_idx],
    params,
    samples,
    progress_directory,
    "predict-bummer-testate.txt")
  save(pred, file = paste0(save_directory, "predict-bummer-testate.RData"))
}
```

S3.4.3 Plot BUMMER predictions

The predictions for 25 randomly chosen hold-out water table depth observations are shown below. The held-out water table depth values are shown in 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.

```
## sorted to increasing values for ease of display
idx <- order(X[sample_idx])
## randomly choose 25 observations to display

n_plot <- 25
n_samples <- length(pred$X[, 1])

sim_df <- data.frame(
  covariate   = c(pred$X[, idx] * sd_X + mean_X),
  mean        = rep(apply(pred$X[, idx] * sd_X + mean_X, 2, mean),
                    each = n_samples),
  median      = rep(apply(pred$X[, idx] * sd_X + mean_X, 2, median),
                    each = n_samples),
  observation = factor(rep(1:n_plot, each = n_samples )),
  truth       = rep(X[sample_idx][idx] * sd_X + mean_X,
                    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="Predicted Water Table Depth (cm)") +
  ggtitle("BUMMER prediction for testate data")
```

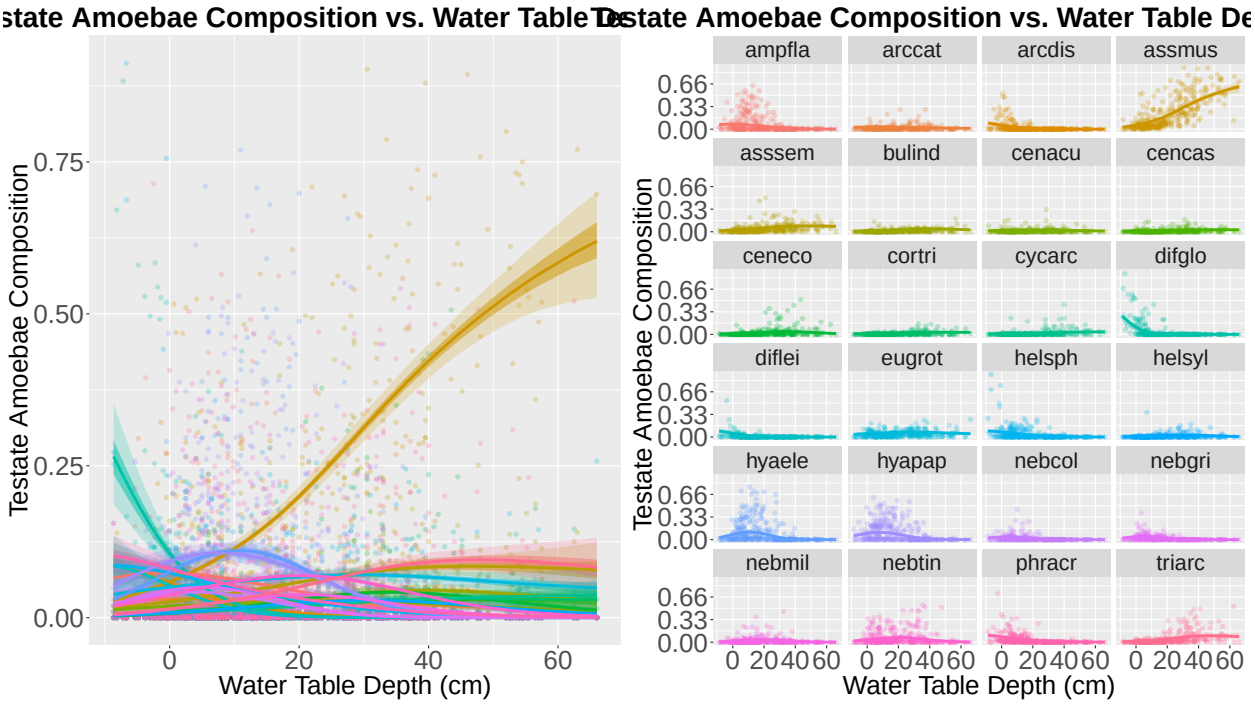


S3.4.4 BUMMER estimated functional response

We also plot the estimated mean functional response with associated Bayesian credible intervals (50% central credible interval is darkly shaded, the 95% central credible interval is lightly shaded).

```
alpha_post_mean <- apply(alpha_post, c(2, 3), mean)
p_alpha_post <- alpha_post
for (k in 1:n_samples) {
  for (i in 1:(N-length(sample_idx))) {
    p_alpha_post[k, i, ] <- alpha_post[k, i, ] / sum(alpha_post[k, i, ])
  }
}

alpha_post_lower_50 <- apply(p_alpha_post, c(2, 3), quantile, prob=0.25)
alpha_post_upper_50 <- apply(p_alpha_post, c(2, 3), quantile, prob=0.75)
alpha_post_lower_95 <- apply(p_alpha_post, c(2, 3), quantile, prob=0.025)
alpha_post_upper_95 <- apply(p_alpha_post, c(2, 3), quantile, prob=0.975)
p_alpha <- apply(p_alpha_post, c(2, 3), mean)
y_prop <- y
for (i in 1:N) {
  y_prop[i, ] <- y[i, ] / sum(y[i, ])
}

fitPlotData <- data.frame(
  species        = factor(rep(colnames(y), each=N-N_pred),
                          levels=colnames(y)),
```

```
  count          = c(y_prop[-sample_idx, ]),
  depth          = rep(X[-sample_idx] * sd_X + mean_X, times=d),
  alpha          = c(p_alpha),
  alpha_lower_50 = c(alpha_post_lower_50),
  alpha_upper_50 = c(alpha_post_upper_50),
  alpha_lower_95 = c(alpha_post_lower_95),
  alpha_upper_95 = c(alpha_post_upper_95))

g1_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  geom_ribbon(aes(ymin=alpha_lower_50, ymax=alpha_upper_50, fill=species, group=species),
              linetype=0, alpha=0.5) +
  geom_ribbon(aes(ymin=alpha_lower_95, ymax=alpha_upper_95, fill=species, group=species),
              linetype=0, alpha=0.2) +
  ggtitle("Testate Amoebae Composition vs. Water Table Depth (cm)") +
  theme(legend.position="none",
        plot.title=element_text(size=26, face="bold", hjust=0.5),
        strip.text.x = element_text(size = 20),
        axis.text.x = element_text(size = 24),
        axis.text.y = element_text(size = 24),
        axis.title.x = element_text(size = 24),
        axis.title.y = element_text(size = 24)) +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25) +
  labs(x="Water Table Depth (cm)", y="Testate Amoebae Composition", size=20)

g2_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  geom_ribbon(aes(ymin=alpha_lower_50, ymax=alpha_upper_50, fill=species, group=species),
              linetype=0, alpha=0.5) +
  geom_ribbon(aes(ymin=alpha_lower_95, ymax=alpha_upper_95, fill=species, group=species),
              linetype=0, alpha=0.2) +
  ggtitle("Testate Amoebae Composition vs. Water Table Depth (cm)") +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25) +
  facet_wrap( ~ species, ncol = 4) +
  labs(x="Water Table Depth (cm)", y="Testate Amoebae Composition", size=20) +
  theme(legend.position="none",
        plot.title=element_text(size=26, face="bold", hjust=0.5),
        strip.text.x = element_text(size = 20),
        axis.text.x = element_text(size = 24),
        axis.text.y = element_text(size = 24),
        axis.title.x = element_text(size = 24),
        axis.title.y = element_text(size = 24)) +
  scale_y_continuous(breaks=c(0.0, 0.33, 0.66, 1.0))

multiplot(g1_post, g2_post, cols=2)
```



S3.5 GAM model fit to the testate amoebae data

The GAM model is a simplification to the full MVGP model. The `BayesComposition` package has an option for fitting the simplified GAM model. We fit the model for 200,000 MCMC iterations, discarding the first 50000 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 is with with a cubic B-spline basis with 6 degrees of freedom and hierarchical pooling priors on the B-spline coefficients $\boldsymbol{\beta}$ using a log-link function.

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

if (file.exists(paste0(save_directory, "fit-gam-testate.RData"))) {
  ## Load MCMC run
  load(paste0(save_directory, "fit-gam-testate.RData"))
} else {
  ##
  ## Long running MCMC
  ##

  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                   = y[ - sample_idx, ],
```

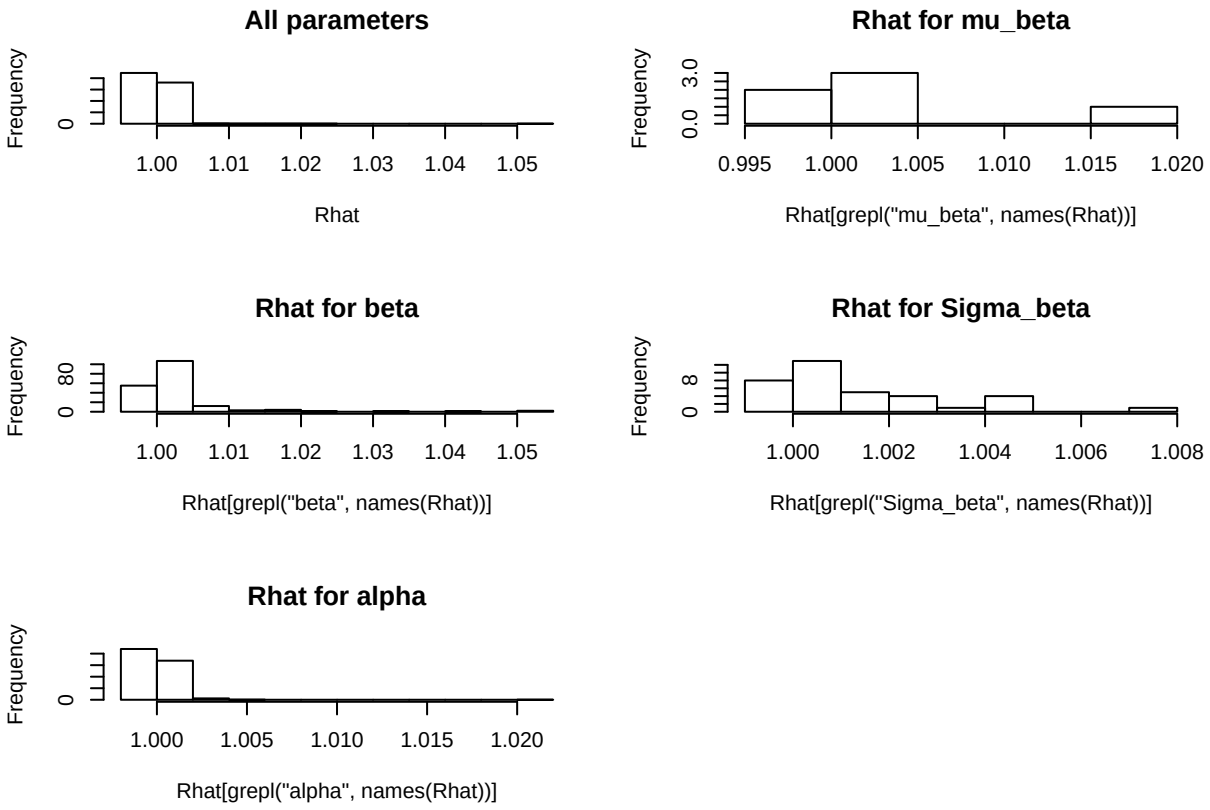
```
    X                  = X[ - sample_idx],
    params             = params,
    progress_directory = progress_directory,
    progress_file      = "fit-gam-testate.txt")

  save(out, sample_idx, file = paste0(save_directory, "fit-gam-testate.RData"))
}
```

S3.5.1 GAM convergence diagnostics

After fitting the GAM 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 $\hat{R} = 1.0521494 \leq 1.1$ at an acceptable value (Gelman et al. [2013], pp.287).

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

n_samples <- nrow(beta_post)

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

```
## [1] 1.052149
```



S3.5.2 Generate GAM predictions

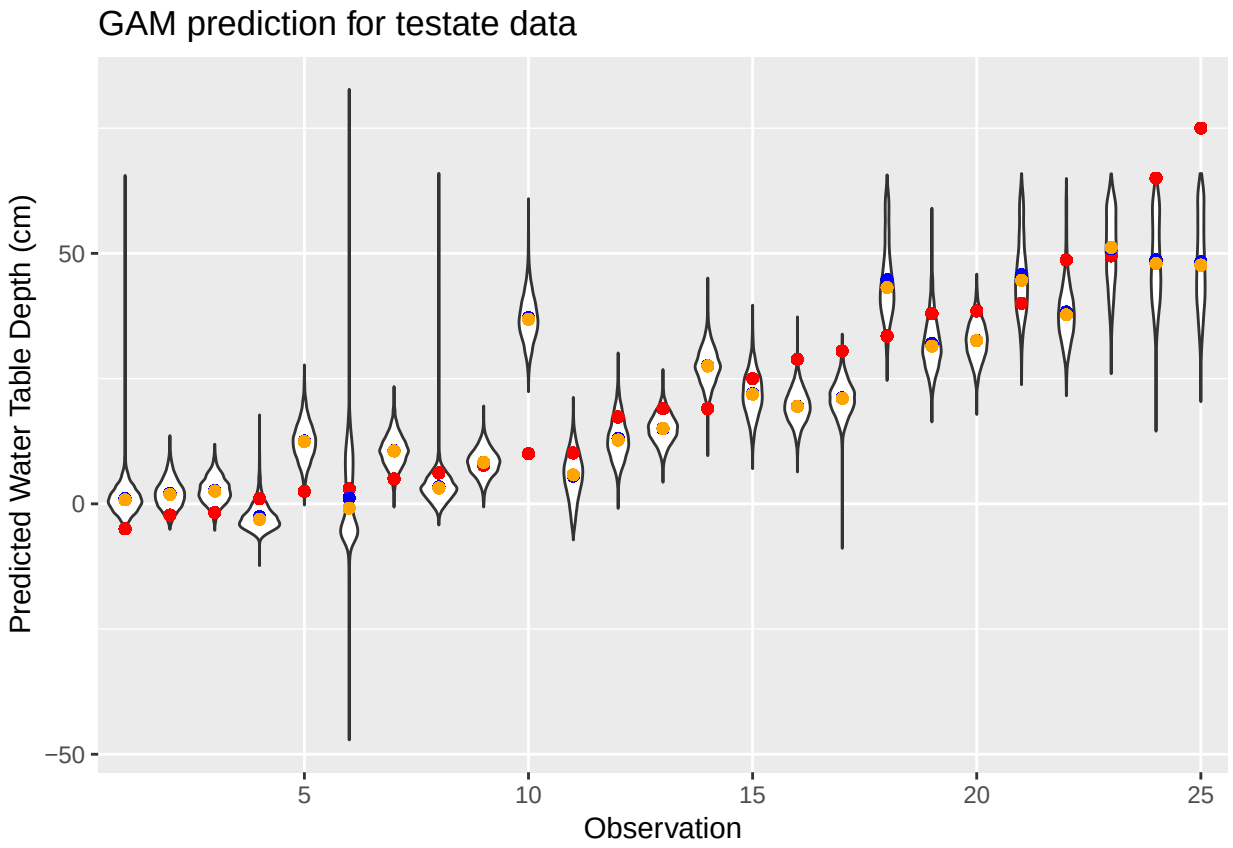
Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved covariates using the posterior samples estimated from the calibration model. These estimates can be generated either jointly with the calibration model or estimated using a two-stage approach 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, "predict-gam-testate.RData"))) {
  load(paste0(save_directory, "predict-gam-testate.RData"))
} else {
  pred <- predict_compositional_data(
    y[sample_idx, ],
    X[ - sample_idx],
    params,
    samples,
    progress_directory,
    "predict-mvgp-testate.txt")
  save(pred, file = paste0(save_directory, "predict-gam-testate.RData"))
}
```

S3.5.3 Plot GAM predictions

The predictions for 25 randomly chosen hold-out water table depth observations are shown below. The held-out values are shown in 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.

```
## sorted to increasing values for ease of display
idx <- order(X[sample_idx])
## randomly choose 25 observations to display

n_plot <- 25
n_samples <- length(pred$X[, 1])

sim_df <- data.frame(
  covariate   = c(pred$X[, idx] * sd_X + mean_X),
  mean        = rep(apply(pred$X[, idx] * sd_X + mean_X, 2, mean),
                    each = n_samples),
  median      = rep(apply(pred$X[, idx] * sd_X + mean_X, 2, median),
                    each = n_samples),
  observation = factor(rep(1:n_plot, each = n_samples )),
  truth       = rep(X[sample_idx][idx] * sd_X + mean_X,
                    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="Predicted Water Table Depth (cm)") +
  ggtitle("GAM prediction for testate data")
```



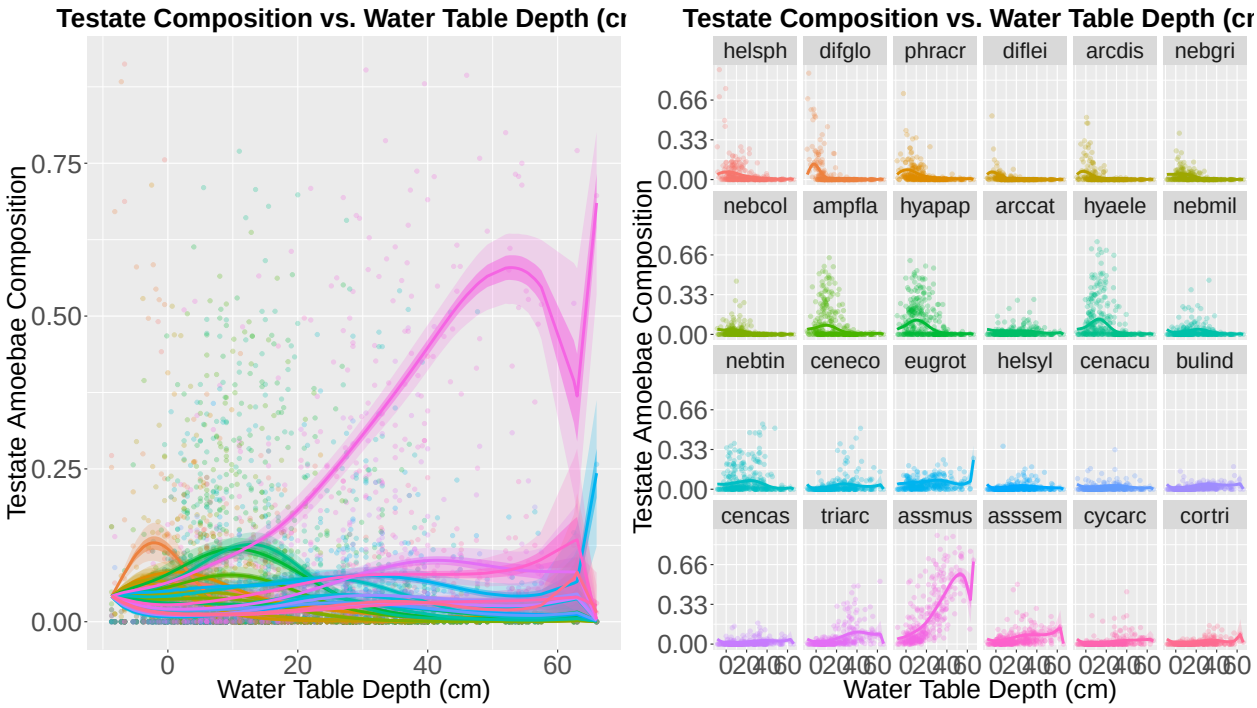
S3.5.4 GAM estimated functional response

We also plot the estimated mean functional response with associated Bayesian credible intervals (50% central credible interval is darkly shaded, the 95% central credible interval is lightly shaded).

```
alpha_post_mean <- apply(alpha_post, c(2, 3), mean)
p_alpha_post <- alpha_post
for (k in 1:n_samples) {
  for (i in 1:(N-length(sample_idx))) {
    p_alpha_post[k, i, ] <- alpha_post[k, i, ] / sum(alpha_post[k, i, ])
  }
}
p_alpha_post_mean <- apply(p_alpha_post, c(2, 3), mean)

alpha_post_lower_50 <- apply(p_alpha_post, c(2, 3), quantile, prob=0.25)
alpha_post_upper_50 <- apply(p_alpha_post, c(2, 3), quantile, prob=0.75)
alpha_post_lower_95 <- apply(p_alpha_post, c(2, 3), quantile, prob=0.025)
alpha_post_upper_95 <- apply(p_alpha_post, c(2, 3), quantile, prob=0.975)

y_prop <- y
for (i in 1:N) {
  y_prop[i, ] <- y[i, ] / sum(y[i, ])
}

fitPlotData <- data.frame(
  species        = factor(rep(colnames(y), each=N-N_pred),
```

```
                          levels=colnames(y)[order_species]),
  count          = c(y_prop[-sample_idx, ]),
  depth          = rep(X[-sample_idx] * sd_X + mean_X, times=d),
  alpha          = c(p_alpha_post_mean),
  alpha_lower_50 = c(alpha_post_lower_50),
  alpha_upper_50 = c(alpha_post_upper_50),
  alpha_lower_95 = c(alpha_post_lower_95),
  alpha_upper_95 = c(alpha_post_upper_95))

g1_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
        plot.title=element_text(size=24, face="bold", hjust=0.5),
        strip.text.x = element_text(size = 20),
        axis.text.x = element_text(size = 24),
        axis.text.y = element_text(size = 24),
        axis.title.x = element_text(size = 24),
        axis.title.y = element_text(size = 24)) +
  geom_ribbon(aes(ymin=alpha_lower_50, ymax=alpha_upper_50, fill=species, group=species),
              linetype=0, alpha=0.5) +
  geom_ribbon(aes(ymin=alpha_lower_95, ymax=alpha_upper_95, fill=species, group=species),
              linetype=0, alpha=0.2) +
  ggtitle("Testate Composition vs. Water Table Depth (cm)") +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25) +
  labs(x="Water Table Depth (cm)", y="Testate Amoebae Composition", size=20)

g2_post <- ggplot(fitPlotData, aes(x=depth, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none",
        plot.title=element_text(size=24, face="bold", hjust=0.5),
        strip.text.x = element_text(size = 20),
        axis.text.x = element_text(size = 24),
        axis.text.y = element_text(size = 24),
        axis.title.x = element_text(size = 24),
        axis.title.y = element_text(size = 24)) +
  geom_ribbon(aes(ymin=alpha_lower_50, ymax=alpha_upper_50, fill=species, group=species),
              linetype=0, alpha=0.5) +
  geom_ribbon(aes(ymin=alpha_lower_95, ymax=alpha_upper_95, fill=species, group=species),
              linetype=0, alpha=0.2) +
  ggtitle("Testate Composition vs. Water Table Depth (cm)") +
  geom_line(aes(x=depth, y=alpha, col = species), fitPlotData, lwd=1.25) +
  facet_wrap( ~ species, ncol = 6) +
  labs(x="Water Table Depth (cm)", y="Testate Amoebae Composition", size=20) +
  scale_y_continuous(breaks=c(0.0, 0.33, 0.66, 1.0))

multiplot(g1_post, g2_post, cols=2)
```

```
## Loading required package: grid
```



S3.6 Cross-validation for the tesate amoebae data

Finally, we provide code for the 12-fold cross-validation experiment. Each of the probabilistic models was fit to the training testate amoebae data for 200,000 MCMC iterations with parameter settings equal to those in the MVGP, BUMMER, and GAM models above. The cross-validation experiment took 22.5 hours running 6 MCMC chains in parallel on a 12-fold cross-validation problem on a 2017 iMac with 4.2GHz processor.

```
y_cv <- y
y_cv_prop <- y_cv

## transform the data to percentages for use in transfer function models
for (i in 1:N) {
  y_cv_prop[i, ] <- y_cv_prop[i, ] / sum(y_cv_prop[i, ])
}
colnames(y_cv) <- as.character(1:dim(y_cv)[2])
colnames(y_cv_prop) <- as.character(1:dim(y_cv)[2])
X_cv <- X
set.seed(11)
# idx <-
kfold <- 12
folds <- cut(sample(1:N, N), breaks=kfold, labels=FALSE)

params <- list(
  n_adapt                    = 50000,
  n_mcmc                     = 150000,
  n_thin                     = 150,
  n_knots                    = n_knots,
  X_knots                    = X_knots,
  message                    = 1000)
```

```
## Setup computing cluster

## define models to fit
models <- c("MVGP", "GAM","BUMMER", "WA", "MAT", "MLRC")

## determines the number of cores on the machine
## make sure you have at least enough cores to fit the model
cps <- 6
## Initalize multicore
sfInit(parallel=TRUE, cpus=cps)
```

```
## R Version:  R version 3.5.2 (2018-12-20)
```

```
## snowfall 1.84-6.1 initialized (using snow 0.4-3): parallel execution on 6 CPUs.
```

```
## Setup random number generator on the cluster
sfClusterSetupRNG()
```

```
## [1] "RNGstream"
```

```
sfLibrary(BayesComposition)
```

```
## Library BayesComposition loaded.
```

```
## Library BayesComposition loaded in cluster.
```

```
## Fit cross-validation
for (model_name in models) {
  if (!file.exists(paste0(save_directory, "cv-testate-", model_name, ".RData"))) {
    message(model_name)
    ##
    ## Long running CV
    ##
    start <- Sys.time()
    if (model_name == "MVGP") {
      ## parameters for MVGP fit
      params$correlation_function <- "exponential"
      params$likelihood <- "dirichlet-multinomial"
      params$function_type <- "gaussian-process"
      params$multiplicative_correlation <- TRUE
      params$additive_correlation <- FALSE

      ## create temporary progress file
      file.create(paste0(progress_directory, "cv-", model_name, "-testate.txt"))

      sink(paste0(progress_directory, "cv-", model_name, "-testate.txt"))
      print(paste("MCMC started at", start))
      sink()
    }
    if (model_name == "BUMMER") {
      ## parameters for BUMMER fit
      params$likelihood <- "dirichlet-multinomial"
      params$function_type <- "bummer"

      ## create temporary progress file
      file.create(paste0(progress_directory, "cv-", model_name, "-testate.txt"))
```

```
    sink(paste0(progress_directory, "cv-", model_name, "-testate.txt"))
    print(paste("MCMC started at", start))
    sink()
  }
  if (model_name == "GAM") {
    ## parameters for GAM fit
    params$likelihood <- "dirichlet-multinomial"
    params$function_type <- "basis"

    ## create temporary progress file
    file.create(paste0(progress_directory, "cv-", model_name, "-testate.txt"))

    sink(paste0(progress_directory, "cv-", model_name, "-testate.txt"))
    print(paste("MCMC started at", start))
    sink()
  }

  sfExport("y_cv", "y_cv_prop", "X_cv", "params", "folds",
           "kfold", "save_directory", "progress_directory",
           "model_name")

  ## Fit the cross-validation function

  CV_out <- sfSapply(
    1:kfold,
    makeCV,
    model_name         = model_name,
    y_cv               = y_cv,
    y_cv_prop          = y_cv_prop,
    X_cv               = X_cv,
    params             = params,
    folds              = folds,
    progress_directory = progress_directory,
    progress_file      = paste0("cv-", model_name, "-testate.txt"),
    dataset            = "testate"
  )

  if (model_name == "MVGP" || model_name == "BUMMER" || model_name == "GAM") {
    ## end timing
    sink(paste0(progress_directory, "cv-", model_name, "-testate.txt"),
         append = TRUE)
    print(Sys.time() - start)
    sink()
  }

  CRPS <- unlist(CV_out[1, ])
  MSPE <- unlist(CV_out[2, ])
  MAE <- unlist(CV_out[3, ])
  coverage <- unlist(CV_out[4, ])
  save(CRPS, MSPE, MAE, coverage,
       file = paste0(save_directory, "cv-testate-", model_name, ".RData"))
 }
}
```

```
## ends snowfall session
sfStop()
```

```
##
## Stopping cluster
```

S3.6.1 Cross-validation results

After running the cross-validation experiment, we extract the model scores and generate a display of cross-validation model performance. For the testate amoebae data in cross-validation, the MAT generates the best point predictions based on MSPE and MAE while MVGP scores second best followed by GAM and WA. MVGP generates the best probabilistic predictions based on CRPS, followed closely by GAM and BUMMER. The Bayesian models do show poor frequentist coverage probabilities suggesting there is work left to better calibrate the Bayesian models.

```
CRPS_out <- matrix(0, length(models), N)
rownames(CRPS_out) <- models
MSPE_out <- matrix(0, length(models), N)
rownames(MSPE_out) <- models
MAE_out <- matrix(0, length(models), N)
rownames(MAE_out) <- models
coverage_out <- matrix(0, length(models), N)
rownames(coverage_out) <- models
idx_model <- 1
for (model_fit in models) {
  ## Load MCMC run
  load(paste0(save_directory, "cv-testate-", model_fit, ".RData"))
  CRPS_out[idx_model, ] <- CRPS
  MSPE_out[idx_model, ] <- MSPE
  MAE_out[idx_model, ] <- MAE
  coverage_out[idx_model, ] <- coverage
  idx_model <- idx_model + 1
}

CRPS <- data.frame(t(apply(CRPS_out, 1, mean, na.rm=TRUE)))
MSPE <- data.frame(t(apply(MSPE_out, 1, mean)))
MAE <- data.frame(t(apply(MAE_out, 1, mean)))
coverage <- data.frame(100/N*t(apply(coverage_out, 1, sum)))

colnames(CRPS) <- models
colnames(MAE) <- models
colnames(MSPE) <- models
colnames(coverage) <- models

cv_results <- t(rbind(CRPS, MSPE, MAE, coverage))
colnames(cv_results) <- c("CRPS", "MSPE", "MAE", "95% CI coverage")
# print(xtable(cv_results, digits=4),
#       file = paste0(save_directory, "cv-results-testate.tex"),
#       floating = FALSE)


kable(cv_results, digits = 4)
```

	CRPS	MSPE	MAE	95% CI coverage
MVGP	0.2959	0.2793	0.3897	78.3708
GAM	0.2966	0.2933	0.3943	79.7753
BUMMER	0.3068	0.3015	0.4069	75.5618
WA	0.4015	0.2924	0.4015	94.3820
MAT	0.3731	0.2476	0.3731	98.5955
MLRC	0.4276	0.4815	0.4276	92.4157

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.

Daniel Lewandowski, Dorota Kurowicka, and Harry Joe. Generating random correlation matrices based on vines and extended onion method. *Journal of Multivariate Analysis*, 100(9):1989–2001, 2009.

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.