Appendix S5: No-analogue Experiment

S5.1 Model performance under no-analogs for testate amoebae data

We conduct a pseudo-analog simulation using the testate data to test the hypothesis that the WA and MAT are sensitive to a lack of analogs. We conduct the experiment by filtering out a proportion of analogs approximately equal in size to a 12-fold cross-validation hold-out size which have the greatest minimum square chord distance from the calibration set. Figure 1 shows the distribution of square-chord distance for the no-analog training and prediction datasets, demonstrating that the training dataset has fewer nearby analogs in the prediction dataset with respect to square chord distance. Using the testate data, we train the model using the closest analogs and predict the unobserved water table depth on the non-analog hold-out data.

The results from the no-analog simulation study in Table 1 show that the proposed MVGP and GAM model show reduced loss of predictive skill under the no analog scenario when compared to WA and MAT. The functional response based methods (MVGP, GAM, and MLRC) do not show as large a decrease in predictive performance because the functional response framework can handle extrapolations to novel compositions more naturally by leveraging the latent functional relationship to the covariate. Modeling the functional response allows for accommodation of non-analog compositions more easily because the interactions among species can be interpolated given the latent response, allowing for prediction using compositions which are not close in square chord distance to the training data. We still see a low empirical coverage rate for MVGP and GAM on the testate amoeba data, suggesting that there is likely overdispersion or other characteristics in the data that are not explained by the MVGP and GAM methods.

S5.2 Model performance under no-analogs for pollen data

We also conduct a pseudo-analog simulation using the pollen data to test the hypothesis that the WA and MAT are sensitive to a lack of analogs. Like the testate amoeba data, we conduct the experiment by filtering out a proportion of analogs approximately equal in size to a 12-fold cross-validation hold-out size. The held-out data are those which have the greatest minimum square chord distance from the calibration set. Figure 2 shows the distribution of square-chord distance for the no-analog training and prediction datasets,

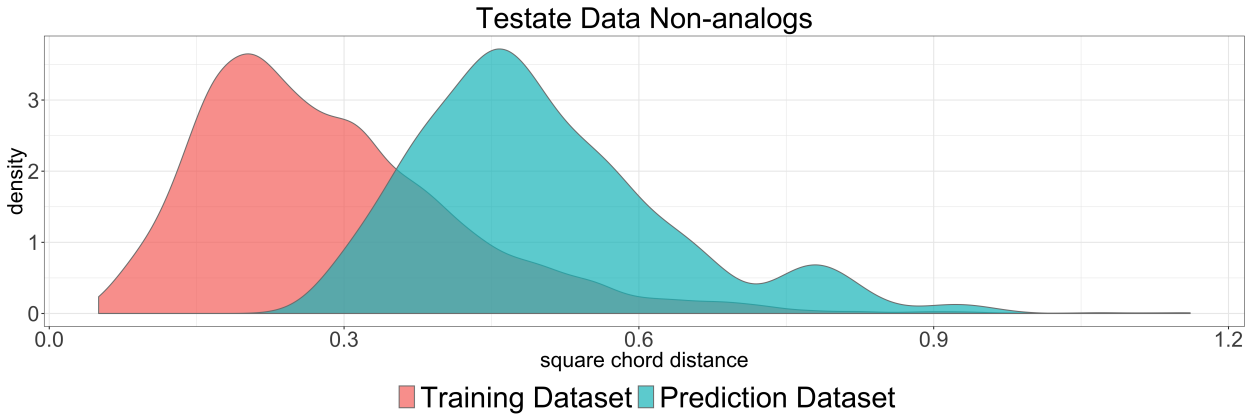


Figure 1: Plot of square chord distance distributions for the training and prediction data for the testate data. The simulated no-analogs square chord distribution is centered at a higher value than the training dataset.

Table 1: Results for predicting unobserved non-analog covariate values using the testate amoeba data. Smaller MSPE, MAE, and CRPS values indicate better model performance. Coverage values closer to the nominal 95% credible interval indicate better model performance.

	CRPS	MSPE	MAE	95% CI coverage
MVGP	0.3673	0.4103	0.4855	68.7500
GAM	0.4115	0.4287	0.5271	62.5000
BUMMER	0.4013	0.4249	0.5052	53.1250
WA	0.5420	0.4995	0.5420	84.3750
MAT	0.5203	0.4498	0.5203	93.7500
MLRC	0.4558	0.4165	0.4558	93.7500

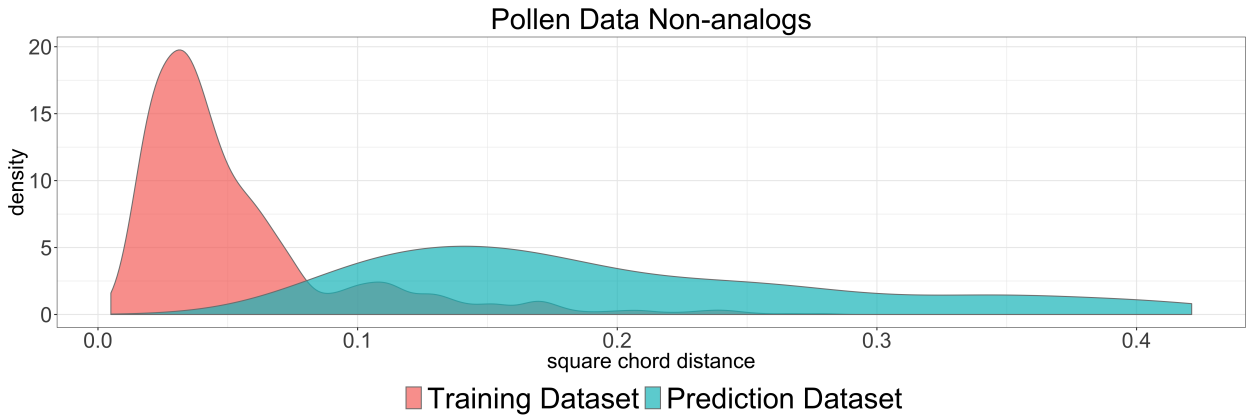


Figure 2: Plot of square chord distance distributions for the training and prediction data for the pollen dataset. The simulated no-analogs square chord distribution is centered at a higher value than the training dataset.

demonstrating that the training dataset has fewer nearby analogs in the prediction dataset with respect to square chord distance. Using the pollen data, we train the model using the closest analogs and predict the unobserved average July Temperature on the non-analog hold-out data.

The results from the no-analog simulation study in Table 2 show that the proposed MVGP and GAM model show reduced loss of predictive skill under the no analog scenario when compared to WA and MAT.

The functional response based methods (MVGP, GAM, and MLRC) do not show as large a decrease in predictive performance because the functional response framework can handle extrapolations to novel compositions more naturally by leveraging the latent functional relationship to the covariate. Modeling the functional response allows for accommodation of non-analog compositions more easily because the interactions among species can be interpolated given the latent response, allowing for prediction using compositions which are not close in square chord distance to the training data. We see a better empirical coverage rate for MVGP and GAM on the pollen data, suggesting that these models are better calibrated to the pollen data.

S5.3 Discussion

Based on the small empirical study presented above, there is evidence that MVGP is less sensitive to a lack of analogs than WA or MAT. We see that the functional response based methods (MVGP, GAM, and MLRC) show less sensitivity to the no-analog problem than the transfer function methods of WA and MAT. The no-analog problem arises in many paleoclimate reconstructions and the implications on the reconstruction are not well understood [Jackson and Williams, 2004]. Modern methods that model the latent functional response like MVGP show evidence of being robust to this problem and further investigation is needed.

Table 2: Results for predicting unobserved non-analog covariate values using the pollen data. Smaller MSPE, MAE, and CRPS values indicate better model performance. Coverage values closer to the nominal 95% credible interval indicate better model performance.

	CRPS	MSPE	MAE	95% CI coverage
MVGP	0.3450	0.3489	0.5022	92.8571
GAM	0.6048	1.0390	0.7998	71.4286
BUMMER	0.6827	1.2655	0.9141	57.1429
WA	0.5373	0.4931	0.5373	78.5714
MAT	0.5245	0.4425	0.5245	100.0000
MLRC	0.5760	0.4964	0.5760	92.8571

S5.4 R code and setup

```
set.seed(11)
library(BayesComposition)
library(knitr)
library(ggplot2)
library(rioja)
library(analogue)
library(here)
library(snowfall)
library(gridExtra)
library(dplyr)
library(here)
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/"
```

S5.5 Testate amoebae data

First we load the testate amoebae data and subset to only include the Booth 2008 data and remove rare species (species with fewer that 500 total counts).

```
# Load Testate Data and R code - Booth 2008
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
```

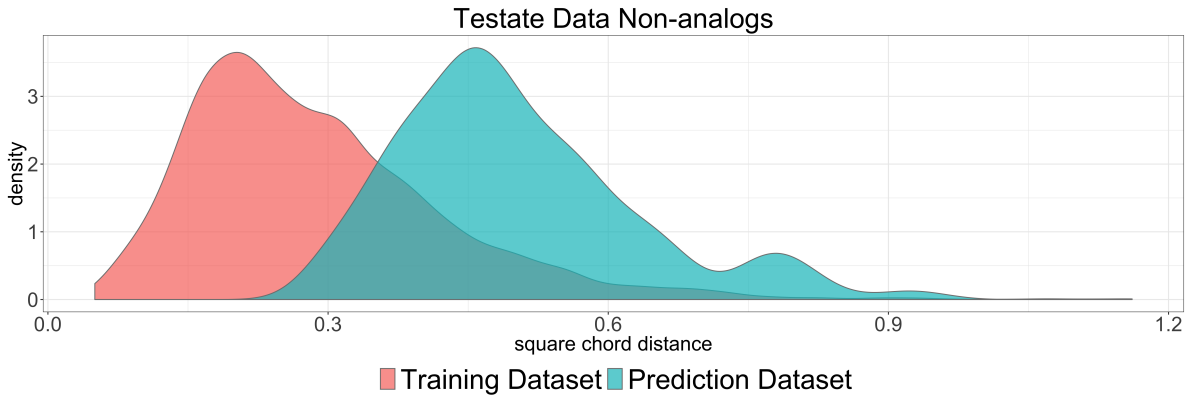
```r
N <- 356
N_obs <- 356
y <- y[1:356, ]
X <- X[1:356]

## join species

y <- join_testate_booth(y)
# source("~/testate/data/join-testate-booth.R")

## remove zeros
no_obs <- which(apply(y, 2, sum) == 0)

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

## Subset rare species, only keeping those with over 500 total count
y <- y[, apply(y, 2, sum) > 500]
# y <- y[, colSums(y) > 100]

mean_X <- mean(X)
sd_X <- sd(X)
X <- (X - mean_X) / sd_X
N <- nrow(y)

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

## 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, ])
}
```

S5.5.1 Generate testate amoebae non-analogs

Next, we generate pseudo-non-analogs by selecting approximately 1/12th of the samples with the largest minimum pairwise square chord distance with all of the other samples. We call these our non-analogs.

```r
## Simulate a "no-analog" situation
modMATDist <- MAT(y_prop, X, k=20, lean=FALSE)

mod_dists <- tibble(set = "mod", distance = as.vector(modMATDist$dist.n))

## 0.275 cut-off chosen so the validation set is approximately the same size
## as one of the 12-fold cross-validation sets
analog_idx <- which(apply(modMATDist$dist.n, 1, min)  > 0.275)
# length(analog_idx) ## 32
# 356/12  ## 29.67

y_train <- y[-analog_idx, ]
y_test <- y[analog_idx, ]
X_train <- X[-analog_idx]
```

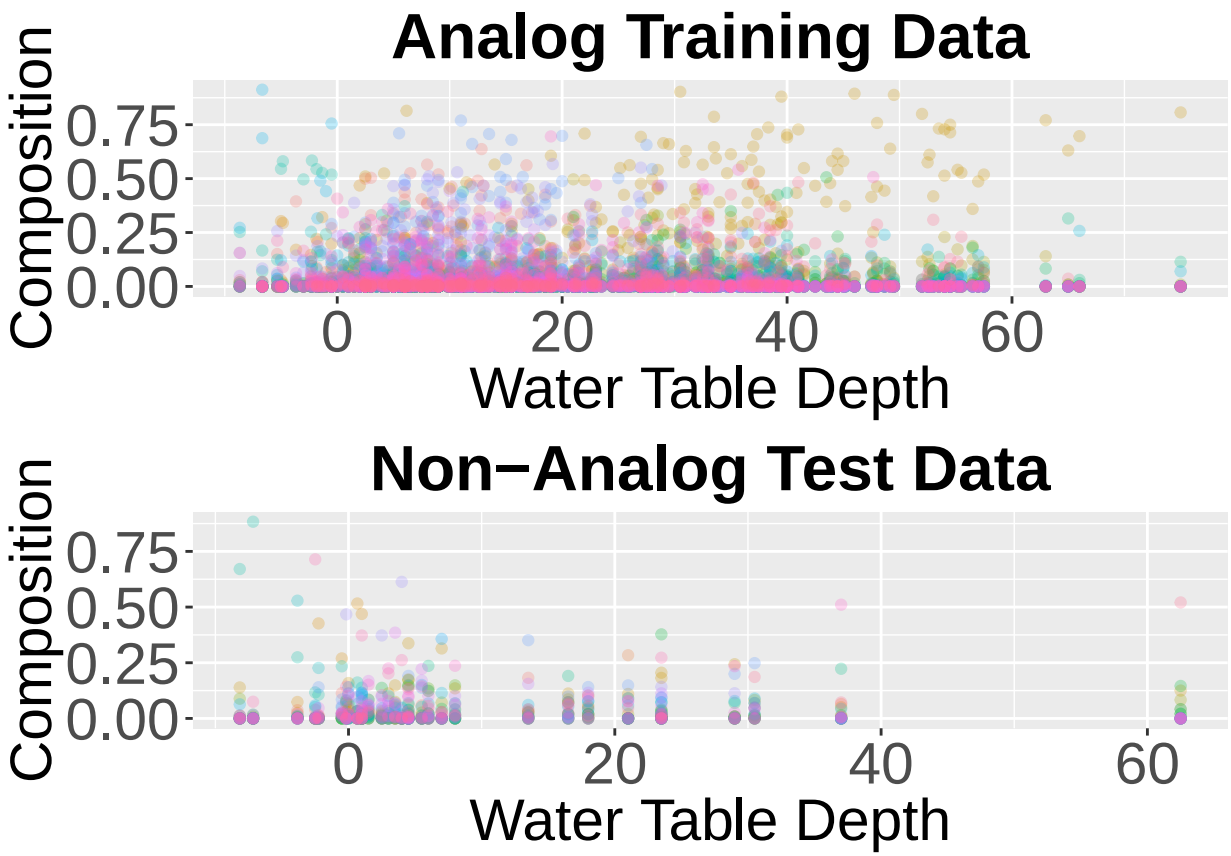
```r
X_test <- X[analog_idx]
y_train_prop <- y_train
for (i in 1:nrow(y_train)) {
  y_train_prop[i, ] <- y_train_prop[i, ] / sum(y_train_prop[i, ])
}
y_test_prop <- y_test
for (i in 1:nrow(y_test)) {
  y_test_prop[i, ] <- y_test_prop[i, ] / sum(y_test_prop[i, ])
}
N_pred <- dim(y_test)[1]

modMATDistTrain <- MAT(y_train_prop, X, k=20, lean=FALSE)
predMATDist <- predict(modMATDistTrain, y_test_prop, k=20, sse=TRUE, n.boot=1000, verbose=FALSE)
mod_dists <- tibble(set = "mod", distance = as.vector(modMATDistTrain$dist.n))
pred_dists <- tibble(set = "pred", distance = as.vector(predMATDist$dist.n))

MAT_dists <- bind_rows(mod_dists, pred_dists)


MAT_dists_plot <- ggplot(MAT_dists, aes(distance, fill=set)) +
  geom_density(alpha=0.7, adjust=1, colour="grey50") +
  labs(x="square chord distance") +
  scale_fill_discrete(name=element_blank(), breaks=c("mod", "pred"),
                      labels=c("Training Dataset", "Prediction Dataset")) +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5, size = 30),
        axis.text.x = element_text(size = 22),
        axis.text.y = element_text(size = 22),
        axis.title.x = element_text(size = 22),
        axis.title.y = element_text(size = 22),
        legend.text=element_text(size=30)) +
    ggtitle("Testate Data Non-analogs")

png(file=paste0(save_directory, "testate-analog-distance.png"),
    width=18, height=6, units="in", res=400)
MAT_dists_plot
invisible(dev.off())
include_graphics(paste0(save_directory, "testate-analog-distance.png"))
```



S5.5.2 Plot the analog and non-analog test data

The training data and the testing (held-out) data are plotted below.

```
analogPlotData <- data.frame(
  species = as.factor(rep(colnames(y), each=length(X_train))),
  Count   = c(as.matrix(y_train_prop)),
  Wetness = rep(X_train * sd_X + mean_X, times=dim(y_train_prop)[2]))

noanalogPlotData <- data.frame(
  species = as.factor(rep(colnames(y), each=length(X_test))),
  Count   = c(as.matrix(y_test_prop)),
  Wetness = rep(X_test * sd_X + mean_X, times=dim(y_test_prop)[2]))

gp1 <- ggplot(analogPlotData, aes(x=Wetness, y=Count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none") + ggtitle("Analog Training Data") +
  labs(x="Water Table Depth", y="Composition") +
  theme(plot.title=element_text(size=24, face="bold", hjust=0.5)) +
  theme(axis.text.x = element_text(size = 22),
        axis.text.y = element_text(size = 22),
        axis.title.x = element_text(size = 22),
        axis.title.y = element_text(size = 22))
gp2 <- ggplot(noanalogPlotData,
              aes(x=Wetness, y=Count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none") + ggtitle("Non-Analog Test Data") +
  labs(x="Water Table Depth", y="Composition") +
  theme(plot.title=element_text(size=24, face="bold", hjust=0.5)) +
  theme(axis.text.x = element_text(size = 22),
        axis.text.y = element_text(size = 22),
        axis.title.x = element_text(size = 22),
        axis.title.y = element_text(size = 22))
multiplot(gp1, gp2, cols=1)
```



S5.5.3 Fitting MVGP to testate amoebae data

We define the parameters for fitting the MVGP model to the testate amoebae data. First we define a sequence of 30 equally-spaced knots over which the predictive process is defined, extending the range of the predictive process knots beyond the range of the data to avoid boundary effects. We have found the reconstructions to be not very sensitive to the choice and spacing of the knots.

```
y <- as.matrix(y)
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)
```

We fit the MVGP model to the testate amoebae data for 200,000 MCMC iterations discarding the first 50,000 iterations as burn-in. We thin every 150 of the remaining 150,000 MCMC samples for 4 parallel chains resulting in 4,000 posterior samples. We fit the MVGP model with an exponential covariance function where we estimate the correlation in functional response. We do not include an overdispersion covariance $\mathbf{\Sigma}_\varepsilon$ in the model. To get the correlation in functional responses to converge, we set the prior $\mathbf{R} \sim \text{LKJ}(\eta = 8)$, providing some shrinkage of the functional correlations towards 0.

```
params <- list(
  n_adapt                    = 50000,
  n_mcmc                     = 150000,
  n_thin                     = 150,
  correlation_function       = "exponential",
  likelihood                 = "dirichlet-multinomial",
  function_type              = "gaussian-process",
  multiplicative_correlation = TRUE,
```

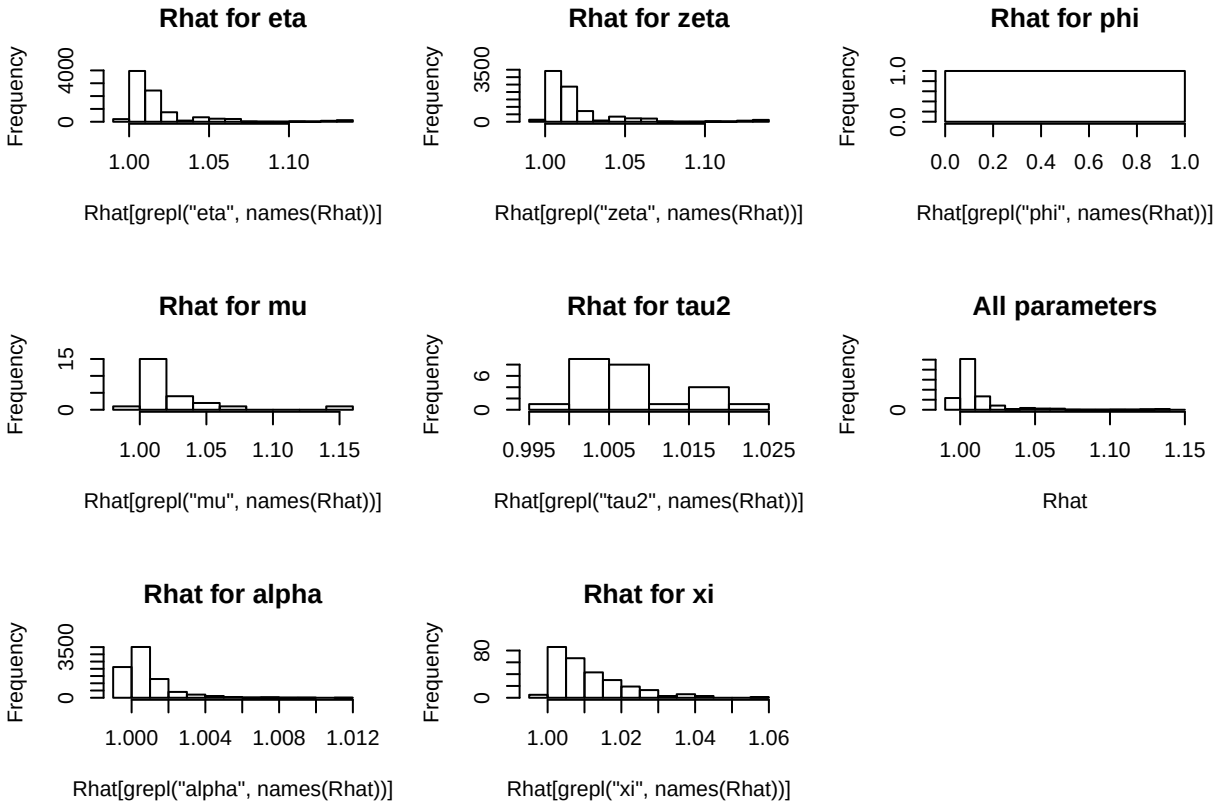
```
  eta                         = 8,
  additive_correlation        = FALSE,
  n_chains                    = 4,
  n_cores                     = 4,
  n_knots                     = n_knots,
  X_knots                     = X_knots)

## Fit no-analog using B-spline model
if (file.exists(paste0(save_directory, "fit-mvgp-testate-no-analog2.RData"))) {

  ## load mcmc
  load(paste0(save_directory, "fit-mvgp-testate-no-analog2.RData"))
} else {

  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                 = y_train,
    X                 = X_train,
    params            = params,
    progress_directory = progress_directory,
    progress_file     = "fit-mvgp-testate-no-analog2.txt")

  save(out, file = paste0(save_directory, "fit-mvgp-testate-no-analog2.RData"))
}
```

S5.5.3.1 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 $\hat{R} = 1.1421732 \leq 1.1$ 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$zeta
alpha_post <- samples$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)


Rhat <- make_gelman_rubin(out)
layout(matrix(1:9, 3, 3))
hist(Rhat[grepl("eta", names(Rhat))], main = "Rhat for eta")
hist(Rhat[grepl("mu", names(Rhat))], main = "Rhat for mu")
hist(Rhat[grepl("alpha", names(Rhat))], main = "Rhat for alpha")
```

```
hist(Rhat[grepl("zeta", names(Rhat))], main = "Rhat for zeta")
hist(Rhat[grepl("tau2", names(Rhat))], main = "Rhat for tau2")
hist(Rhat[grepl("xi", names(Rhat))], main = "Rhat for xi")
hist(Rhat[grepl("phi", names(Rhat))], main = "Rhat for phi")
hist(Rhat, main="All parameters")
```



S5.5.3.2 Generating MVGP predictions

Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved water table depth using the posterior samples estimated from the testate amoebae 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.

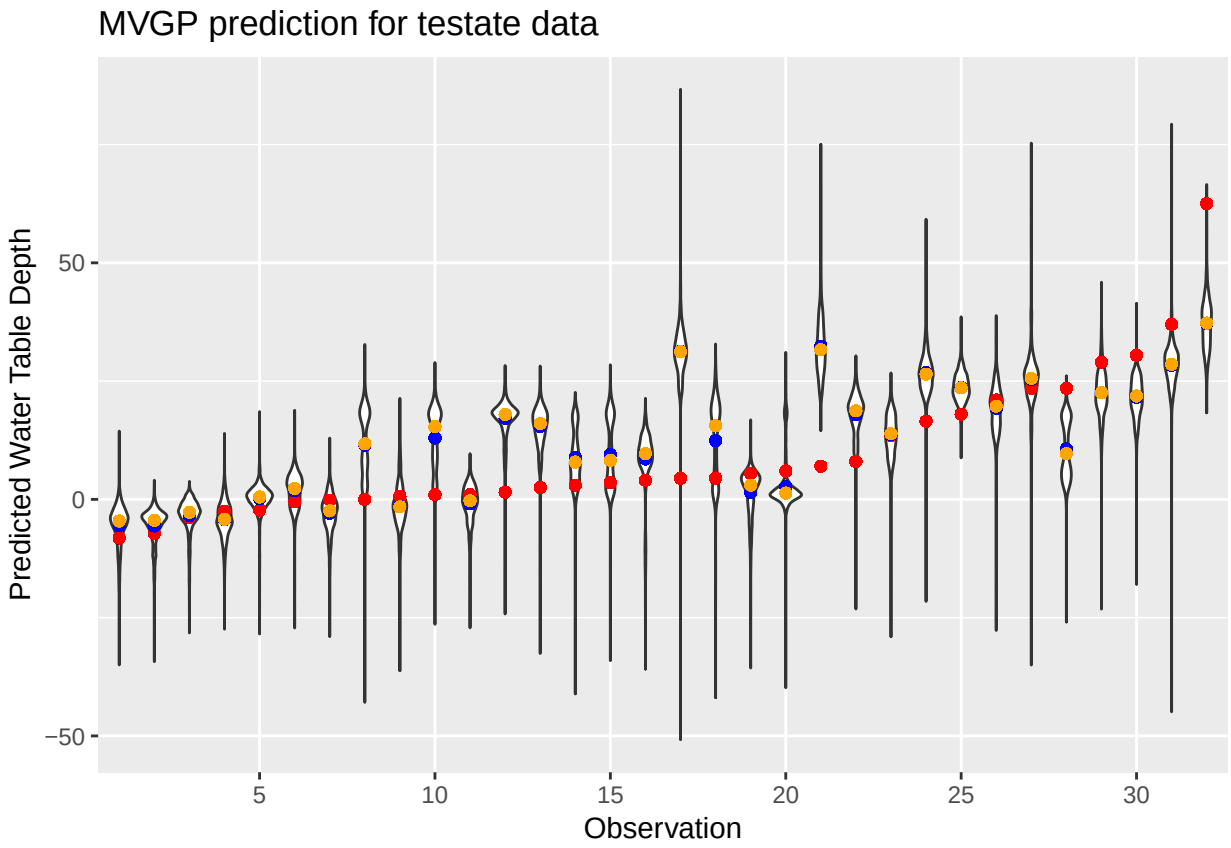
```
## Note, there might be some initial warnings about ESS shrunk to the
##    current position for the first few iterations. This is not a
##    major concern as long as the warnings don't continue.
if (file.exists(paste0(save_directory, "predict-mvgp-testate-no-analog2.RData"))) {
  load(paste0(save_directory, "predict-mvgp-testate-no-analog2.RData"))
} else {
  pred <- predict_compositional_data(
    y_test,
    X_train,
    params,
    samples,
    progress_directory,
```

```
    "predict-mvgp-testate-no-analog2.txt")
  save(pred, file = paste0(save_directory, "predict-mvgp-testate-no-analog2.RData"))
}
```

S5.5.3.3 Plot MVGP predictions

The predictions for the chosen non-analog missing covariate observations are shown below. The held-out covariate 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 non-analog water table depth covariate.

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

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], 2, median) * sd_X + mean_X,
                    each = n_samples),
  observation = factor(rep(1:length(analog_idx), each = n_samples )),
  truth       = rep(X_test[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, length(analog_idx), 5)) +
  labs(x="Observation", y="Predicted Water Table Depth") +
  ggtitle("MVGP prediction for testate data")
```



S5.5.4 GAM model fit to testate amoebae data

We fit the GAM model for 200,000 MCMC iterations discarding the first 50,000 iterations as burn-in. We thin every 150 of the remaining 150,000 MCMC samples for 4 parallel chains resulting in 4,000 posterior samples. We used the default setting in `BayesComposition` resulting in a cubic B-spline model with 6 degrees of freedom.

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

## Fit no-analog using B-spline model
if (file.exists(paste0(save_directory, "fit-gam-testate-no-analog.RData"))) {
  ## load mcmc
  load(paste0(save_directory, "fit-gam-testate-no-analog.RData"))
} else {

  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                  = y_train,
    X                  = X_train,
```

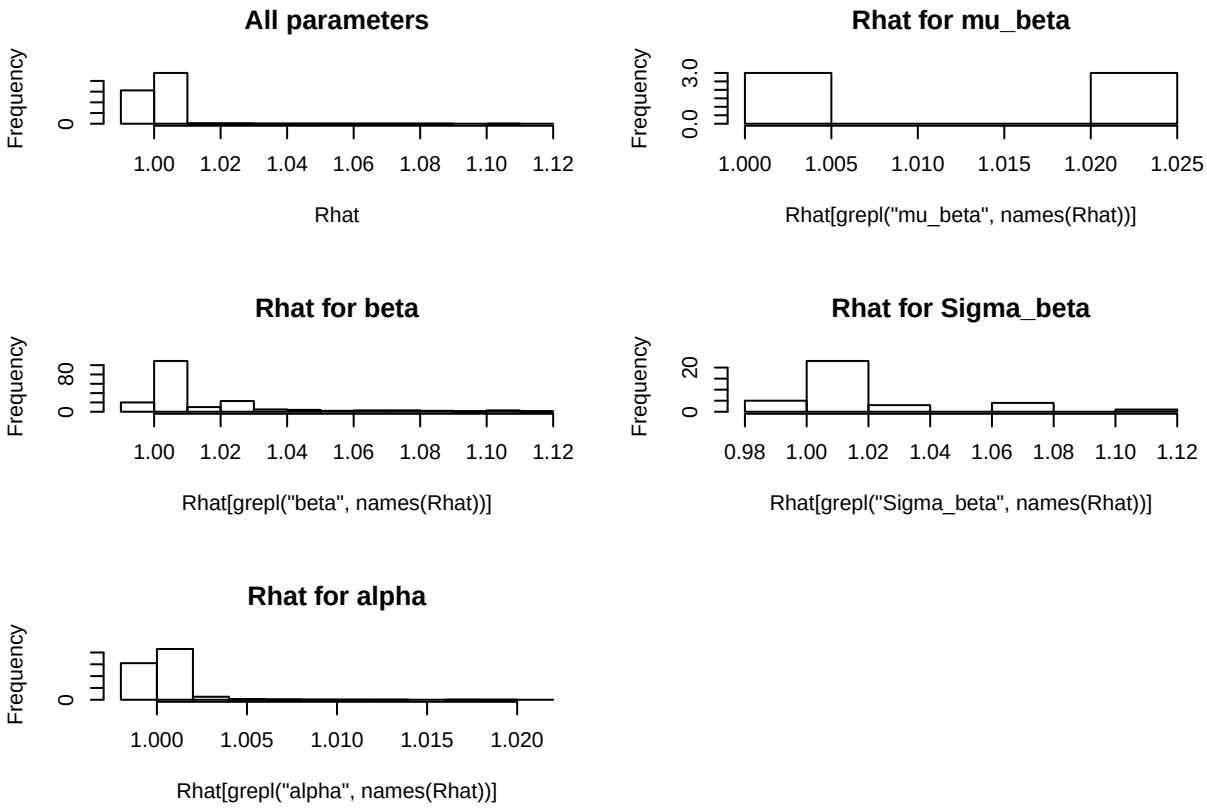
```
    params              = params,
    progress_directory = progress_directory,
    progress_file       = "fit-gam-testate-no-analog.txt")

  save(out, file = paste0(save_directory, "fit-gam-testate-no-analog.RData"))
}
```

S5.5.4.1 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 $\hat{R} = 1.1148867 \leq 1.1$ 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] 1.114887
```



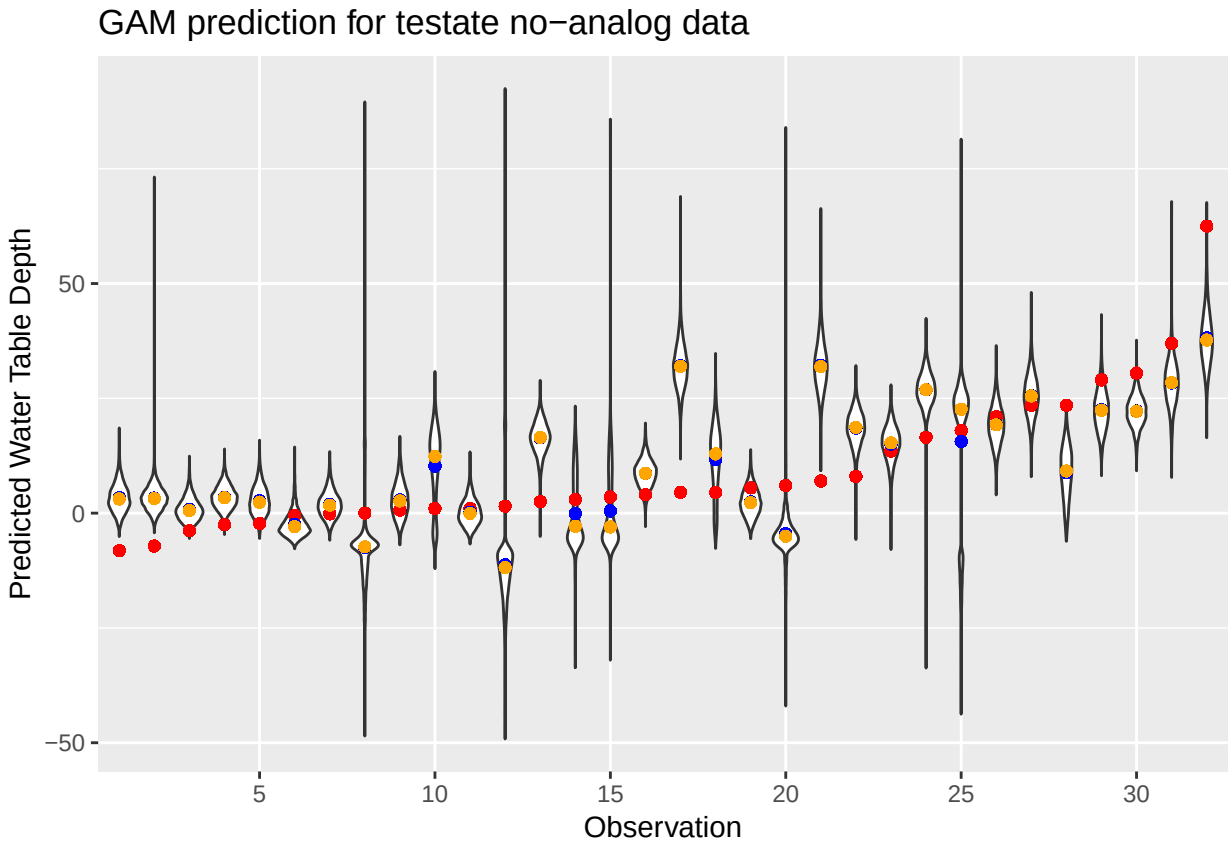
S5.5.4.2 Generate GAM predictions

```
## 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-no-analog.RData"))) {
  load(paste0(save_directory, "predict-gam-testate-no-analog.RData"))
} else {
  pred_gam <- predict_compositional_data(
    y_test,
    X_train,
    params,
    samples,
    progress_directory,
    "predict-gam-testate-no-analog.txt")
  save(pred_gam, file = paste0(save_directory, "predict-gam-testate-no-analog.RData"))
}
```

S5.5.4.3 Plot GAM predictions

The predictions for the chosen non-analog missing covariate observations are shown below. The held-out covariate 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 not reconstructing the unobserved as well as the MVGP model.

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

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

sim_df <- data.frame(
  covariate   = c(pred_gam$X[, idx] * sd_X + mean_X),
  mean        = rep(apply(pred_gam$X[, idx] * sd_X + mean_X, 2, mean),
                    each = n_samples),
  median      = rep(apply(pred_gam$X[, idx], 2, median) * sd_X + mean_X,
                    each = n_samples),
  observation = factor(rep(1:length(analog_idx), each = n_samples )),
  truth       = rep(X_test[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, length(analog_idx), 5)) +
  labs(x="Observation", y="Predicted Water Table Depth") +
  ggtitle("GAM prediction for testate no-analog data")
```



S5.5.5 Bummer model fit to testate amoebae data

We fit the BUMMER model for 200,000 MCMC iterations discarding the first 50,000 iterations as burn-in. We thin every 150 of the remaining 150,000 MCMC samples for 4 parallel chains resulting in 4,000 posterior samples.

```
params <- list(
  n_adapt                    = 50000,
  n_mcmc                     = 150000,
  n_thin                     = 150,
  likelihood                 = "dirichlet-multinomial",
  function_type              = "bummer",
  n_chains                   = 4,
  n_cores                    = 4)


## Fit no-analog using bummer model
if (file.exists(paste0(save_directory, "fit-bummer-testate-no-analog.RData"))) {

  ## load mcmc
  load(paste0(save_directory, "fit-bummer-testate-no-analog.RData"))
} else {

  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                  = y_train,
    X                  = X_train,
    params             = params,
    progress_directory = progress_directory,
    progress_file      = "fit-bummer-testate-no-analog.txt")

  save(out, file = paste0(save_directory, "fit-bummer-testate-no-analog.RData"))
}
```

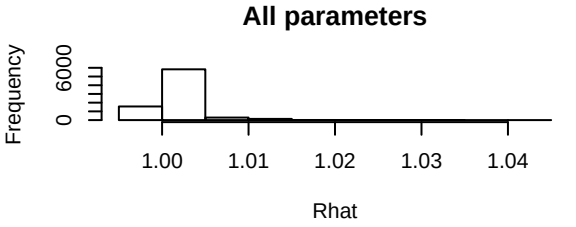
S5.5.5.1 BUMMER convergence diagnostics

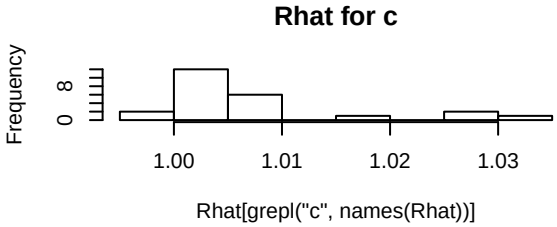
After fitting the model, the posterior samples are generated and MCMC convergence is assessed using the Gelman-Rubin $\hat{R}$ statistic. We see the distribution of $\hat{R}$ statistics is close to 1 with the largest $\hat{R} = 1.0436622 \leq 1.1$ at an acceptable value (Gelman et al. [2013], pp.287).

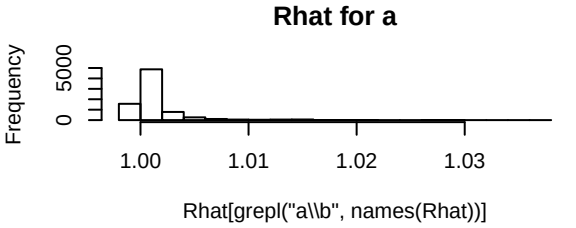
```
## extract posterior samples
samples <- extract_compositional_samples(out)
a_post <- samples$a
b_post <- samples$b
c_post <- samples$c
alpha_post <- samples$alpha

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")
```

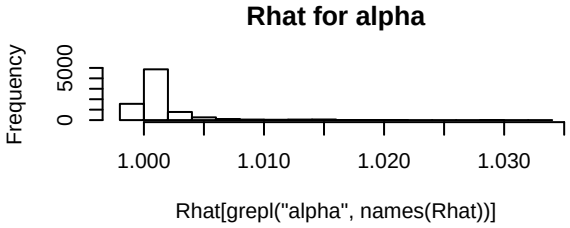
```
hist(Rhat[grepl("b", names(Rhat))], main = "Rhat for b")
hist(Rhat[grepl("c", names(Rhat))], main = "Rhat for c")
hist(Rhat[grepl("alpha", names(Rhat))], main = "Rhat for alpha")
Rhat[!is.finite(Rhat)] <- NA
max(unlist(na.omit(Rhat)))
```

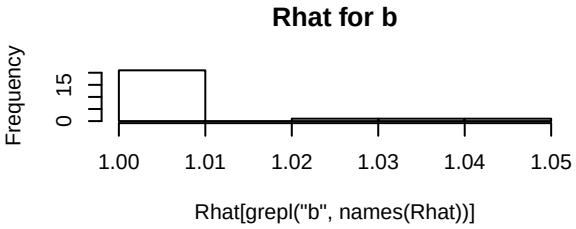
```
## [1] 1.043662
```











S5.5.5.2 Generate BUMMER predictions

Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved water table depth using the posterior samples estimated from the testate amoebae 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.

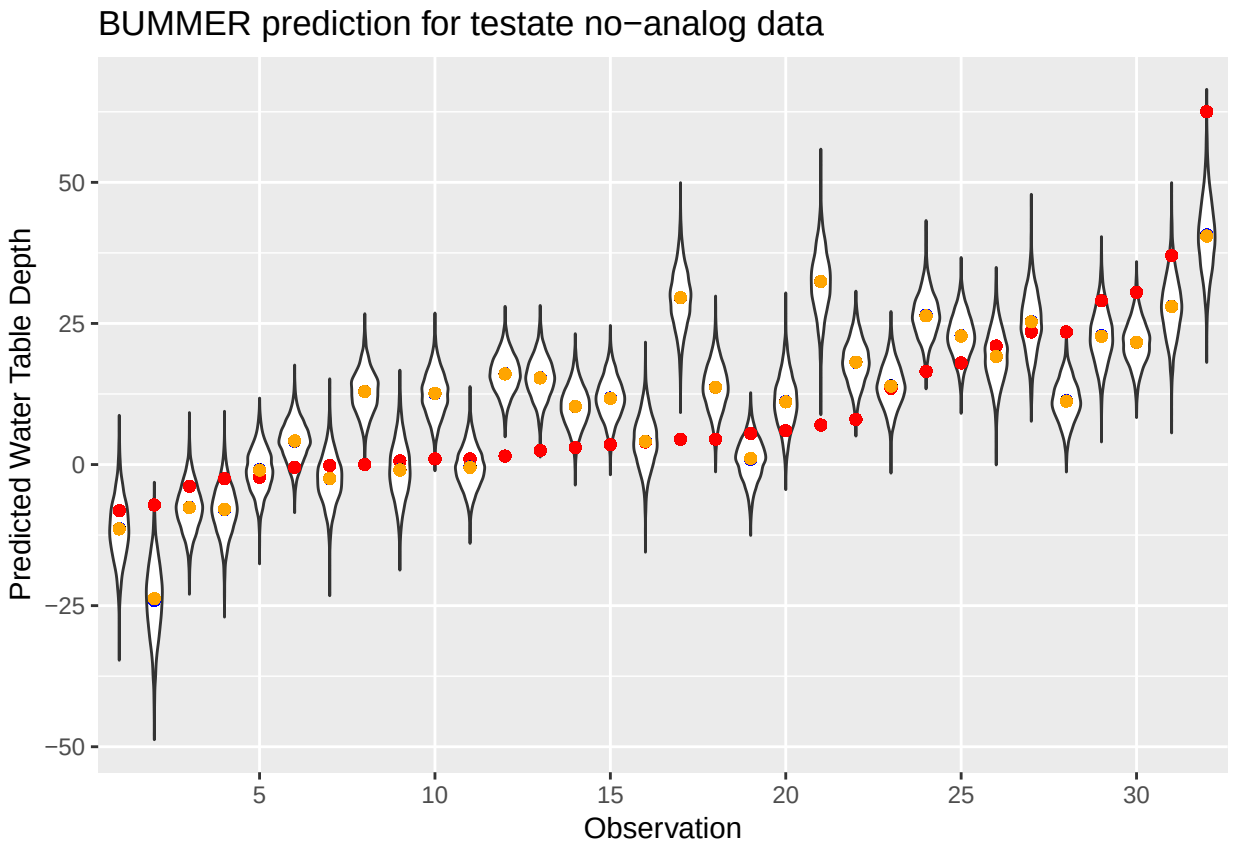
```
## Note, there might be some initial warnings about ESS shrunk to the
##    current position for the first few iterations. This is not a
##    major concern as long as the warnings don't continue.
if (file.exists(paste0(save_directory, "predict-bummer-testate-no-analog.RData"))) {
  load(paste0(save_directory, "predict-bummer-testate-no-analog.RData"))
} else {
  pred_bummer <- predict_compositional_data(
    y_test,
    X_train,
    params,
    samples,
```

```
    progress_directory,
    "predict-bummer-testate-no-analog.txt")
  save(pred_bummer, file = paste0(save_directory, "predict-bummer-testate-no-analog.RData"))
}
```

S5.5.5.3 Plot BUMMER predictions

The predictions for the chosen non-analog missing water table depth observations are shown below. The held-out covariate 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 not reconstructing the held-out test data as well as the MVGP model.

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

n_samples <- length(pred_bummer$X[, 1])

sim_df <- data.frame(
  covariate   = c(pred_bummer$X[, idx] * sd_X + mean_X),
  mean        = rep(apply(pred_bummer$X[, idx] * sd_X + mean_X, 2, mean),
                    each = n_samples),
  median      = rep(apply(pred_bummer$X[, idx], 2, median) * sd_X + mean_X,
                    each = n_samples),
  observation = factor(rep(1:length(analog_idx), each = n_samples )),
  truth       = rep(X_test[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, length(analog_idx), 5)) +
  labs(x="Observation", y="Predicted Water Table Depth") +
  ggtitle("BUMMER prediction for testate no-analog data")
```



S5.5.6 Fitting transfer function models to the non-analog testate amoebae data

Below, we fit the transfer function methods WA, MAT, and MLRC using the `R rioja` package.

S5.5.6.1 Generate non-analog predictive scores

After fitting all of the models under consideration, we compute the predictive scores for the testate amoebae data below.

```
models <- c("MVGP", "GAM", "BUMMER", "WA", "MAT", "MLRC")
n_models <- length(models)
N_no_analog <- length(analog_idx)
coverage <- matrix(0, N_no_analog, n_models)
MSPE <- matrix(0, N_no_analog, n_models)
MAE <- matrix(0, N_no_analog, n_models)
CRPS <- matrix(0, N_no_analog, n_models)

MSPE[, 1] <- (X_test - apply(pred$X, 2, mean))^2
MSPE[, 2] <- (X_test - apply(pred_gam$X, 2, mean))^2
MSPE[, 3] <- (X_test - apply(pred_bummer$X, 2, mean))^2
MSPE[, 4] <- (X_test - pred_mu_WA)^2
MSPE[, 5] <- (X_test - pred_mu_MAT)^2
MSPE[, 6] <- (X_test - pred_mu_MLRC)^2

MAE[, 1] <- abs(X_test - apply(pred$X, 2, median))
MAE[, 2] <- abs(X_test - apply(pred_gam$X, 2, median))
```

```
MAE[, 3] <- abs(X_test - apply(pred_bummer$X, 2, median))
MAE[, 4] <- abs(X_test - pred_mu_WA)
MAE[, 5] <- abs(X_test - pred_mu_MAT)
MAE[, 6] <- abs(X_test - pred_mu_MLRC)

coverage[, 1] <-
  (X_test > apply(pred$X, 2, quantile, prob=0.025)) &
  (X_test < apply(pred$X, 2, quantile, prob=0.975))
coverage[, 2] <-
  (X_test > apply(pred_gam$X, 2, quantile, prob=0.025)) &
  (X_test < apply(pred_gam$X, 2, quantile, prob=0.975))
coverage[, 3] <-
  (X_test > apply(pred_bummer$X, 2, quantile, prob=0.025)) &
  (X_test < apply(pred_bummer$X, 2, quantile, prob=0.975))
coverage[, 4] <-
  (X_test > (pred_mu_WA - 2 * pred_sd_WA)) &
  (X_test < (pred_mu_WA + 2 * pred_sd_WA))
coverage[, 5] <-
  (X_test > (pred_mu_MAT - 2 * pred_sd_MAT)) &
  (X_test < (pred_mu_MAT + 2 * pred_sd_MAT))
coverage[, 6] <-
  (X_test > (pred_mu_MLRC - 2 * pred_sd_MLRC)) &
  (X_test < (pred_mu_MLRC + 2 * pred_sd_MLRC))

CRPS[, 1] <- makeCRPS(pred$X, X_test, dim(pred$X)[1])
CRPS[, 2] <- makeCRPS(pred_gam$X, X_test, dim(pred_gam$X)[1])
CRPS[, 3] <- makeCRPS(pred_bummer$X, X_test, dim(pred_bummer$X)[1])
CRPS[, 4] <- MAE[, 4]
CRPS[, 5] <- MAE[, 5]
CRPS[, 6] <- MAE[, 6]

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


results <- rbind(
  apply(CRPS, 2, mean), apply(MSPE, 2, mean),
  apply(MAE, 2, mean), 100*apply(coverage, 2, mean))
rownames(results) <- c("CRPS", "MSPE", "MAE", "95% CI coverage")
# print(xtable(t(results), digits=4),
#       file=here("results", "appendix-booth.tex"),
#       floating=FALSE)
```

```
kable(t(results), digits = 4, format = "latex", longtable = FALSE)
```

	CRPS	MSPE	MAE	95% CI coverage
MVGP	0.3673	0.4103	0.4855	68.750
GAM	0.4115	0.4287	0.5271	62.500
BUMMER	0.4013	0.4249	0.5052	53.125
WA	0.5420	0.4995	0.5420	84.375
MAT	0.5203	0.4498	0.5203	93.750
MLRC	0.4558	0.4165	0.4558	93.750

S5.6 Non-analog Pollen Data

First, we load the data.

```
dat <- read.csv(here::here("data", "Reduced.Taxa.calibration.3.23.17.csv"),
                stringsAsFactors=FALSE, header=TRUE)
N <- length(dat$ACERX[-1])
d <- 16
y <- matrix(c(as.numeric(dat$ACERX[-1]), as.numeric(dat$BETULA[-1]),
              as.numeric(dat$Sum.Other.Conifer[-1]), as.numeric(dat$LARIXPSEU[-1]),
              as.numeric(dat$Sum.Other.Deciduous[-1]), as.numeric(dat$FAGUS[-1]),
              as.numeric(dat$FRAXINUX[-1]), as.numeric(dat$Sum.Other.Herbaceous[-1]),
              as.numeric(dat$Sum.Prairie.Herbs[-1]), as.numeric(dat$Other[-1]),
              as.numeric(dat$PICEAX[-1]), as.numeric(dat$PINUSX[-1]),
              as.numeric(dat$QUERCUS[-1]), as.numeric(dat$TILIA[-1]),
              as.numeric(dat$TSUGAX[-1]), as.numeric(dat$ULMUS[-1])), N, d)

## Adjust for half counts
y <- ceiling(y)
colnames(y) <- names(dat)[5:20]

X_annual <- as.numeric(dat$tmean_annual[-1])
X <- as.numeric(dat$tmean_07[-1])

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, ])
}
```

S5.6.1 Generate pollen non-analogs

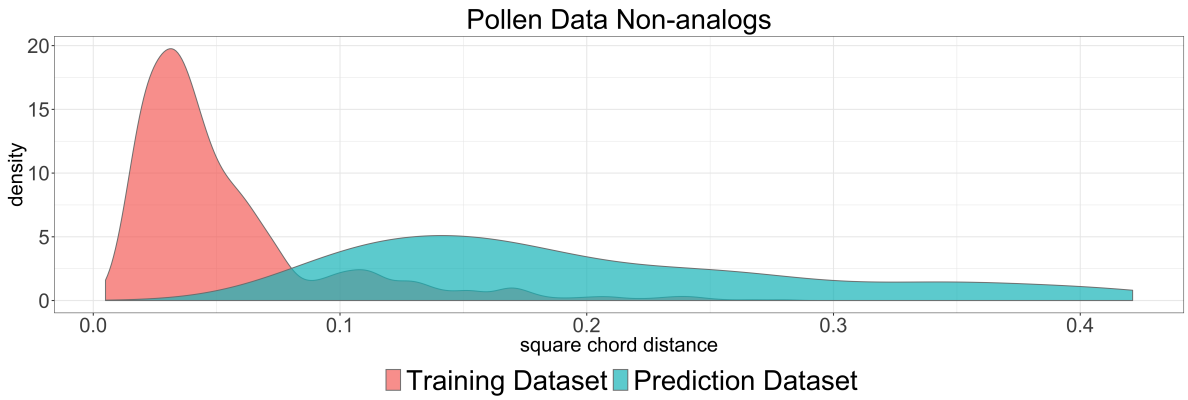
Next, we generate pseudo-non-analogs by selecting approximately 1/12th of the samples with the largest minimum pairwise square chord distance with all of the other samples. We call these our non-analogs.

```
## Simulate a "no-analog" situation
modMATDist <- MAT(as.data.frame(y_prop), X, k=10, lean=FALSE)

mod_dists <- tibble(set = "mod", distance = as.vector(modMATDist$dist.n))

## 0.008 cut-off chosen so the validation set is approximately the same size
## as one of the 12-fold cross-validation sets
analog_idx <- which(apply(modMATDist$dist.n, 1, min)  > 0.08)
# length(analog_idx) ## 14
# 152/12  ## 12.67

y_train <- y[-analog_idx, ]
y_test <- y[analog_idx, ]
X_train <- X[-analog_idx]
X_test <- X[analog_idx]
```

```r
y_train_prop <- y_train
for (i in 1:nrow(y_train)) {
  y_train_prop[i, ] <- y_train_prop[i, ] / sum(y_train_prop[i, ])
}
y_test_prop <- y_test
for (i in 1:nrow(y_test)) {
  y_test_prop[i, ] <- y_test_prop[i, ] / sum(y_test_prop[i, ])
}
N_pred <- dim(y_test)[1]


modMATDistTrain <- MAT(as.data.frame(y_train_prop), X_train, k=10, lean=FALSE)
predMATDist <- predict(modMATDistTrain, as.data.frame(y_test_prop),
                       k=10, sse=TRUE, n.boot=1000)
```

```
## Bootstrap sample 50
## Bootstrap sample 100
```

```r
mod_dists <- tibble(set = "mod", distance = as.vector(modMATDistTrain$dist.n))
pred_dists <- tibble(set = "pred", distance = as.vector(predMATDist$dist.n))

MAT_dists <- bind_rows(mod_dists, pred_dists)


MAT_dists_plot <- ggplot(MAT_dists, aes(distance, fill=set)) +
  geom_density(alpha=0.7, adjust=1, colour="grey50") +
  labs(x="square chord distance") +
  scale_fill_discrete(name=element_blank(), breaks=c("mod", "pred"),
                      labels=c("Training Dataset", "Prediction Dataset")) +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5, size = 30),
        axis.text.x = element_text(size = 22),
        axis.text.y = element_text(size = 22),
        axis.title.x = element_text(size = 22),
        axis.title.y = element_text(size = 22),
        legend.text=element_text(size=30)) +
  ggtitle("Pollen Data Non-analogs")


png(file=paste0(save_directory, "pollen-analog-distance.png"),
    width=18, height=6, units="in", res=400)
MAT_dists_plot
invisible(dev.off())
include_graphics(paste0(save_directory, "pollen-analog-distance.png"))
```



S5.6.2 MVGP model fit to pollen non-analog data.

We define the parameters for fitting the MVGP model to the pollen data. First we define a sequence of 30 equally-spaced knots over which the predictive process is defined, extending the range of the predictive process knots beyond the range of the data to avoid boundary effects. We have found the reconstructions to be not very sensitive to the choice and spacing of the knots.

```
y <- as.matrix(y)
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)
```

We fit the MVGP model to the pollen data for 200,000 MCMC iterations discarding the first 50,000 iterations as burn-in. We thin every 150 of the remaining 150,000 MCMC samples for 4 parallel chains resulting in 4,000 posterior samples. We fit the MVGP model with an exponential covariance function where we estimate the correlation in functional response. We do not include an overdispersion covariance $\mathbf{\Sigma}_\varepsilon$ in the model.

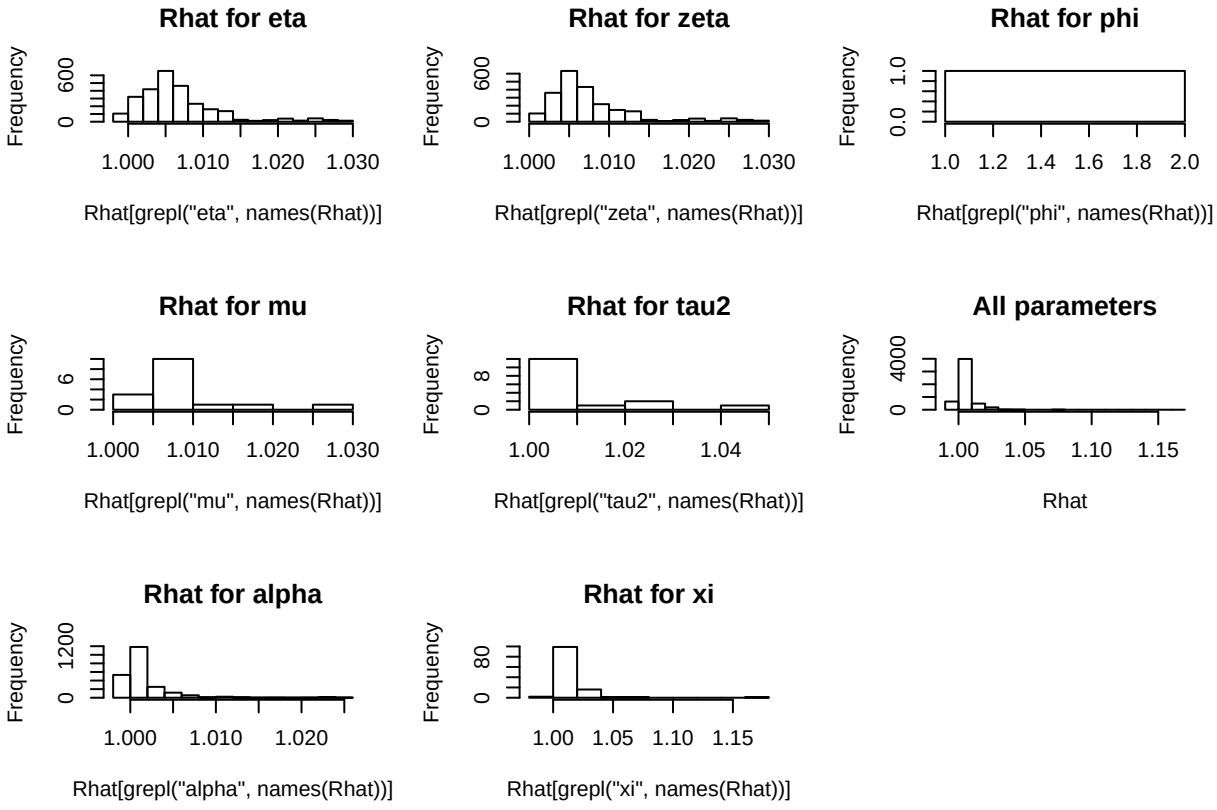
```
params <- list(
  n_adapt                    = 50000,
  n_mcmc                     = 150000,
  n_thin                     = 150,
  correlation_function       = "exponential",
  likelihood                 = "dirichlet-multinomial",
  function_type              = "gaussian-process",
  multiplicative_correlation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

## Fit no-analog using MVGP model
if (file.exists(paste0(save_directory, "fit-mvgp-pollen-no-analog2.RData"))) {

  ## load mcmc
  load(paste0(save_directory, "fit-mvgp-pollen-no-analog2.RData"))
} else {

  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                  = y_train,
```

```
    X                 = X_train,
    params            = params,
    progress_directory = progress_directory,
    progress_file     = "fit-mvgp-pollen-no-analog2.txt")



  save(out, file = paste0(save_directory, file = "fit-mvgp-pollen-no-analog2.RData"))
}
```

S5.6.2.1 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 $\hat{R} = 1.1619608 \leq 1.1$ 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$zeta
alpha_post <- samples$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)

Rhat <- make_gelman_rubin(out)
layout(matrix(1:9, 3, 3))
hist(Rhat[grepl("eta", names(Rhat))], main = "Rhat for eta")
hist(Rhat[grepl("mu", names(Rhat))], main = "Rhat for mu")
hist(Rhat[grepl("alpha", names(Rhat))], main = "Rhat for alpha")
hist(Rhat[grepl("zeta", names(Rhat))], main = "Rhat for zeta")
hist(Rhat[grepl("tau2", names(Rhat))], main = "Rhat for tau2")
hist(Rhat[grepl("xi", names(Rhat))], main = "Rhat for xi")
hist(Rhat[grepl("phi", names(Rhat))], main = "Rhat for phi")
hist(Rhat, main="All parameters")
```



S5.6.2.2 Generating MVGP predictions

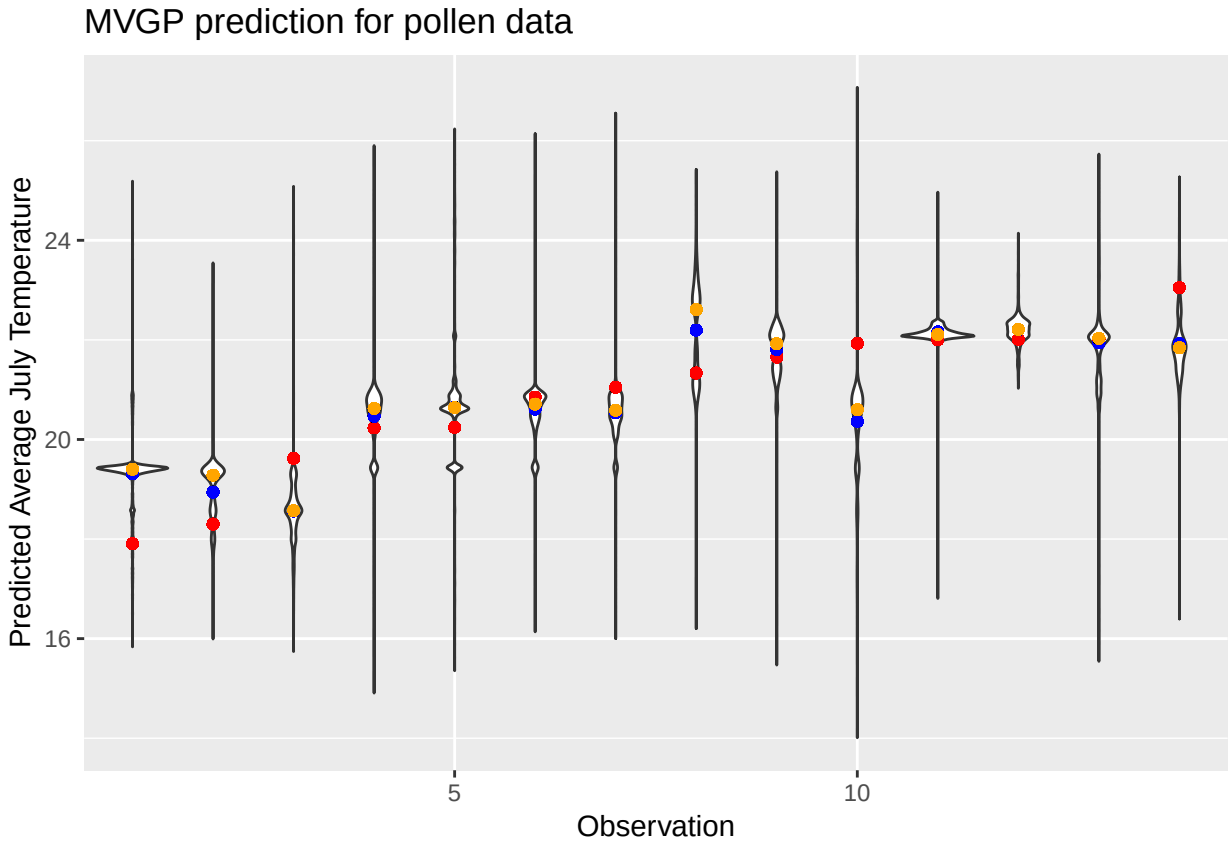
Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved average July temperature depth using the posterior samples estimated from the testate amoebae calibration model. These estimates can be generated either jointly with the calibration model or 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-pollen-no-analog2.RData"))) {
  load(paste0(save_directory, "predict-mvgp-pollen-no-analog2.RData"))
} else {
  pred <- predict_compositional_data(
    y_test,
    X_train,
    params,
    samples,
    progress_directory,
    "predict-mvgp-pollen-no-analog2.txt")
  save(pred, file = paste0(save_directory, "predict-mvgp-pollen-no-analog2.RData"))
}
```

S5.6.2.3 Plot MVGP predictions

The predictions for the chosen non-analog missing covariate observations are shown below. The held-out covariate 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_test)
## randomly choose 25 observations to display

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], 2, median) * sd_X + mean_X,
                    each = n_samples),
  observation = factor(rep(1:length(analog_idx), each = n_samples )),
  truth       = rep(X_test[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, length(analog_idx), 5)) +
  labs(x="Observation", y="Predicted Average July Temperature") +
  ggtitle("MVGP prediction for pollen data")
```



S5.6.3 GAM model fit

We fit the GAM model for 200,000 MCMC iterations discarding the first 50,000 iterations as burn-in. We thin every 150 of the remaining 150,000 MCMC samples for 4 parallel chains resulting in 4,000 posterior samples. We used the default setting in `BayesComposition` resulting in a cubic B-spline model with 6 degrees of freedom.

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

## Fit no-analog using B-spline model
if (file.exists(paste0(save_directory, "fit-gam-pollen-no-analog.RData"))) {
  ## load mcmc
  load(paste0(save_directory, "fit-gam-pollen-no-analog.RData"))
} else {

  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                 = y_train,
    X                 = X_train,
```

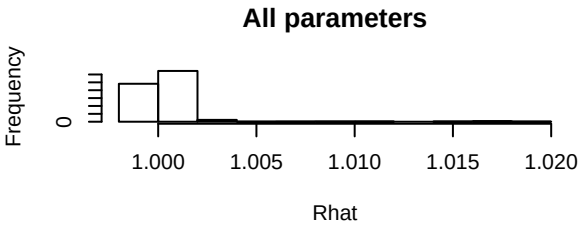
```
    params              = params,
    progress_directory = progress_directory,
    progress_file      = "fit-gam-pollen-no-analog.txt")

  save(out, file = paste0(save_directory, "fit-gam-pollen-no-analog.RData"))
}
```

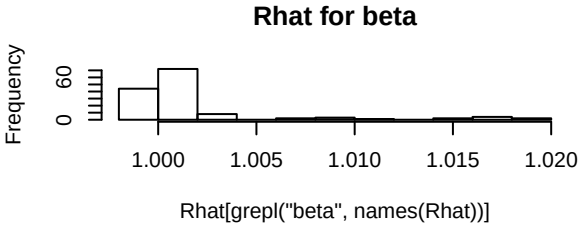
S5.6.3.1 GAM convergence diagnostics

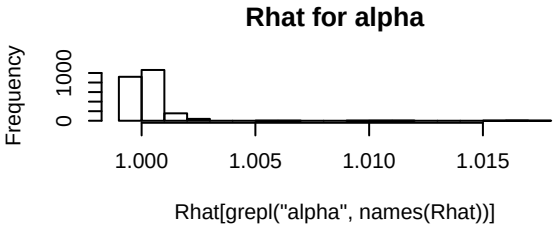
After fitting the model, the posterior samples are generated and MCMC convergence is assessed using the Gelman-Rubin $\hat{R}$ statistic. We see the distribution of $\hat{R}$ statistics is close to 1 with the largest $\hat{R} = 1.018457 \leq 1.1$ at an acceptable value (Gelman et al. [2013], pp.287).

```
## extract posterior samples
samples <- extract_compositional_samples(out)
beta_post <- samples$beta
alpha_post <- samples$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")
Rhat[!is.finite(Rhat)] <- NA
max(unlist(na.omit(Rhat)))
```

```
## [1] 1.018457
```







S5.6.3.2 Generating GAM predictions

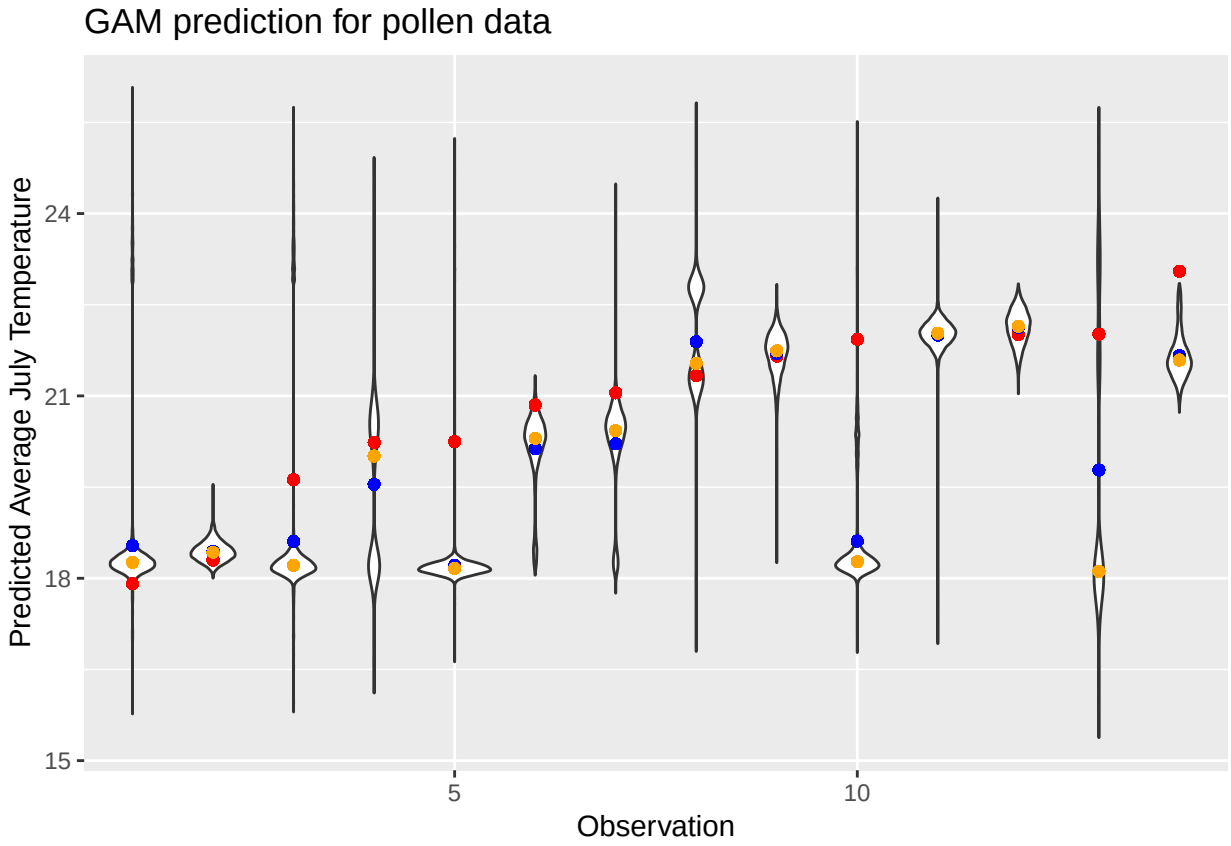
Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved average July temperature depth using the posterior samples estimated from the testate amoebae calibration model. These estimates can be generated either jointly with the calibration model or 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-pollen-no-analog.RData"))) {
  load(paste0(save_directory, "predict-gam-pollen-no-analog.RData"))
} else {
  pred_gam <- predict_compositional_data(
    y_test,
    X_train,
    params,
    samples,
    progress_directory,
    "predict-gam-pollen-no-analog.txt")
  save(pred_gam, file = paste0(save_directory, "predict-gam-pollen-no-analog.RData"))
}
```

S5.6.3.3 Plot GAM predictions

The predictions for the chosen non-analog missing covariate observations are shown below. The held-out covariate 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 not doing as well at estimating the held-out non-analogs as the MVGP model.

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

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

sim_df <- data.frame(
  covariate   = c(pred_gam$X[, idx] * sd_X + mean_X),
  mean        = rep(apply(pred_gam$X[, idx] * sd_X + mean_X, 2, mean),
                    each = n_samples),
  median      = rep(apply(pred_gam$X[, idx], 2, median) * sd_X + mean_X,
                    each = n_samples),
  observation = factor(rep(1:length(analog_idx), each = n_samples )),
  truth       = rep(X_test[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, length(analog_idx), 5)) +
  labs(x="Observation", y="Predicted Average July Temperature") +
  ggtitle("GAM prediction for pollen data")
```



S5.6.4 BUMMER model fit to testate amoebae data

We fit the BUMMER model for 200,000 MCMC iterations discarding the first 50,000 iterations as burn-in. We thin every 150 of the remaining 150,000 MCMC samples for 4 parallel chains resulting in 4,000 posterior samples.

```
params <- list(
  n_adapt                    = 50000,
  n_mcmc                     = 150000,
  n_thin                     = 150,
  likelihood                 = "dirichlet-multinomial",
  function_type              = "bummer",
  n_chains                   = 4,
  n_cores                    = 4)


## Fit no-analog using bummer model
if (file.exists(paste0(save_directory, "fit-bummer-pollen-no-analog.RData"))) {

  ## load mcmc
  load(paste0(save_directory, "fit-bummer-pollen-no-analog.RData"))
} else {

  ## potentially long running MCMC code
  out <- fit_compositional_data(
    y                  = y_train,
```

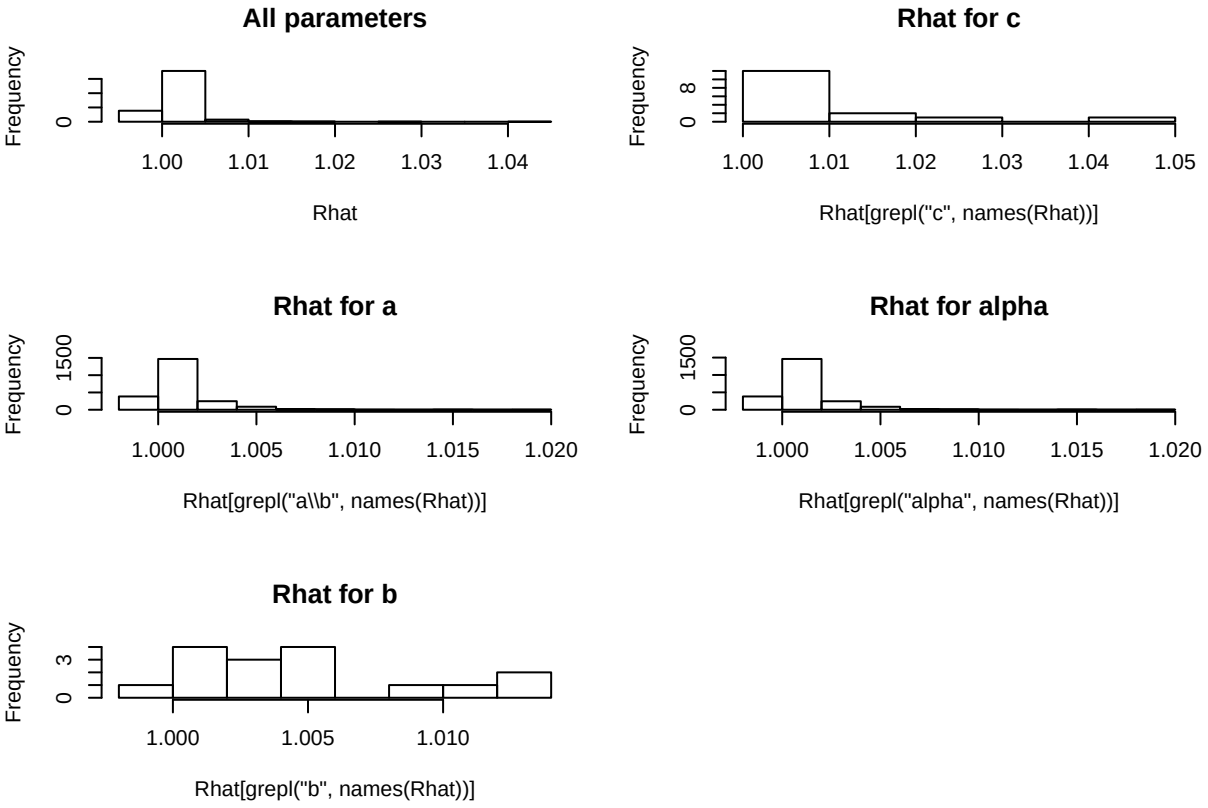
```
    X                 = X_train,
    params            = params,
    progress_directory = progress_directory,
    progress_file     = "fit-bummer-pollen-no-analog.txt")

  save(out, file = paste0(save_directory, "fit-bummer-pollen-no-analog.RData"))
}
```

S5.6.4.1 BUMMER 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 $\hat{R} = 1.0411651 \leq 1.1$ 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] 1.041165
```



S5.6.4.2 Generating BUMMER predictions

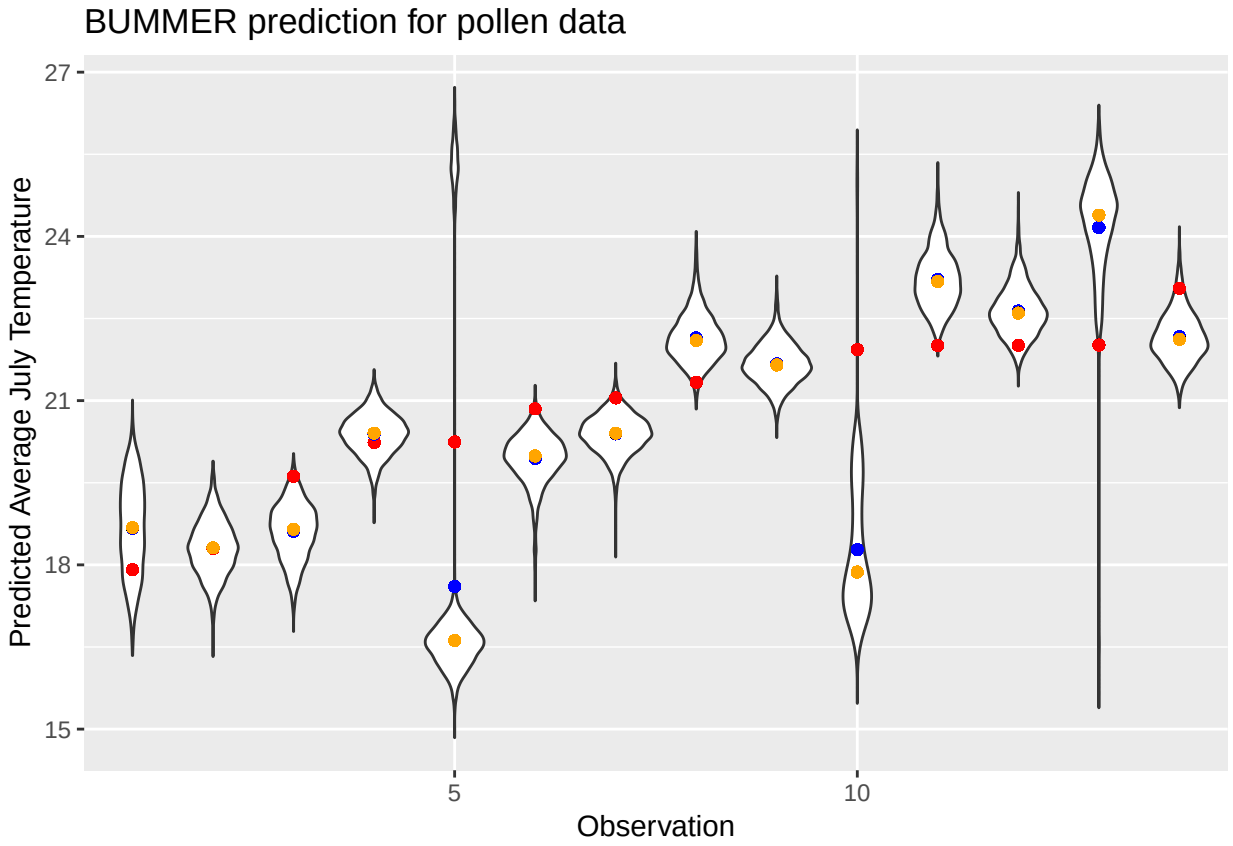
Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved average July temperature depth using the posterior samples estimated from the testate amoebae calibration model. These estimates can be generated either jointly with the calibration model or 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-bummer-pollen-no-analog.RData"))) {
  load(paste0(save_directory, "predict-bummer-pollen-no-analog.RData"))
} else {
  pred_bummer <- predict_compositional_data(
    y_test,
    X_train,
    params,
    samples,
    progress_directory,
    "predict-bummer-pollen-no-analog.txt")
  save(pred_bummer, file = paste0(save_directory, "predict-bummer-pollen-no-analog.RData"))
}
```

S5.6.4.3 Plot BUMMER predictions

The predictions for the chosen non-analog missing covariate observations are shown below. The held-out covariate 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 not doing as well at estimating the held-out non-analogs as the MVGP model.

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

n_samples <- length(pred_bummer$X[, 1])

sim_df <- data.frame(
  covariate   = c(pred_bummer$X[, idx] * sd_X + mean_X),
  mean        = rep(apply(pred_bummer$X[, idx] * sd_X + mean_X, 2, mean),
                    each = n_samples),
  median      = rep(apply(pred_bummer$X[, idx], 2, median) * sd_X + mean_X,
                    each = n_samples),
  observation = factor(rep(1:length(analog_idx), each = n_samples )),
  truth       = rep(X_test[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, length(analog_idx), 5)) +
  labs(x="Observation", y="Predicted Average July Temperature") +
  ggtitle("BUMMER prediction for pollen data")
```



S5.6.5 Fitting transfer function models to the non-analog testate amoebae data

Below, we fit the transfer function methods WA, MAT, and MLRC using the `R rioja` package.

S5.6.5.1 Generate non-analog predictive scores

After fitting all of the models under consideration, we compute the predictive scores for the pollen data below.

```
models <- c("MVGP", "GAM", "BUMMER", "WA", "MAT", "MLRC")
n_models <- length(models)
N_no_analog <- length(analog_idx)
coverage <- matrix(0, N_no_analog, n_models)
MSPE <- matrix(0, N_no_analog, n_models)
MAE <- matrix(0, N_no_analog, n_models)
CRPS <- matrix(0, N_no_analog, n_models)

MSPE[, 1] <- (X_test - apply(pred$X, 2, mean))^2
MSPE[, 2] <- (X_test - apply(pred_gam$X, 2, mean))^2
MSPE[, 3] <- (X_test - apply(pred_bummer$X, 2, mean))^2
MSPE[, 4] <- (X_test - pred_mu_WA)^2
MSPE[, 5] <- (X_test - pred_mu_MAT)^2
MSPE[, 6] <- (X_test - pred_mu_MLRC)^2

MAE[, 1] <- abs(X_test - apply(pred$X, 2, median))
MAE[, 2] <- abs(X_test - apply(pred_gam$X, 2, median))
MAE[, 3] <- abs(X_test - apply(pred_bummer$X, 2, median))
```

```
MAE[, 4] <- abs(X_test - pred_mu_WA)
MAE[, 5] <- abs(X_test - pred_mu_MAT)
MAE[, 6] <- abs(X_test - pred_mu_MLRC)

coverage[, 1] <-
  (X_test > apply(pred$X, 2, quantile, prob=0.025)) &
  (X_test < apply(pred$X, 2, quantile, prob=0.975))
coverage[, 2] <-
  (X_test > apply(pred_gam$X, 2, quantile, prob=0.025)) &
  (X_test < apply(pred_gam$X, 2, quantile, prob=0.975))
coverage[, 3] <-
  (X_test > apply(pred_bummer$X, 2, quantile, prob=0.025)) &
  (X_test < apply(pred_bummer$X, 2, quantile, prob=0.975))
coverage[, 4] <-
  (X_test > (pred_mu_WA - 2 * pred_sd_WA)) &
  (X_test < (pred_mu_WA + 2 * pred_sd_WA))
coverage[, 5] <-
  (X_test > (pred_mu_MAT - 2 * pred_sd_MAT)) &
  (X_test < (pred_mu_MAT + 2 * pred_sd_MAT))
coverage[, 6] <-
  (X_test > (pred_mu_MLRC - 2 * pred_sd_MLRC)) &
  (X_test < (pred_mu_MLRC + 2 * pred_sd_MLRC))


CRPS[, 1] <- makeCRPS(pred$X, X_test, dim(pred$X)[1])
CRPS[, 2] <- makeCRPS(pred_gam$X, X_test, dim(pred_gam$X)[1])
CRPS[, 3] <- makeCRPS(pred_bummer$X, X_test, dim(pred_bummer$X)[1])
CRPS[, 4] <- MAE[, 4]
CRPS[, 5] <- MAE[, 5]
CRPS[, 6] <- MAE[, 6]

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


results <- rbind(
  apply(CRPS, 2, mean), apply(MSPE, 2, mean),
  apply(MAE, 2, mean), 100*apply(coverage, 2, mean))
rownames(results) <- c("CRPS", "MSPE", "MAE", "95% CI coverage")

# print(xtable(t(results), digits=4),
#       file=here("results", "appendix-pollen.tex"),
#       floating=FALSE)
```

```
kable(t(results), digits = 4, format = "latex", longtable = FALSE)
```

	CRPS	MSPE	MAE	95% CI coverage
MVGP	0.3450	0.3489	0.5022	92.8571
GAM	0.6048	1.0390	0.7998	71.4286
BUMMER	0.6827	1.2655	0.9141	57.1429
WA	0.5373	0.4931	0.5373	78.5714
MAT	0.5245	0.4425	0.5245	100.0000
MLRC	0.5760	0.4964	0.5760	92.8571

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.

Stephen T Jackson and John W Williams. Modern analogs in Quaternary paleoecology: Here today, gone yesterday, gone tomorrow? *Annual Review Earth Planetary Science*, 32:495–537, 2004.

Iain Murray, Ryan Prescott Adams, and David J. C. MacKay. Elliptical slice sampling. In *AISTATS*, volume 13, pages 541–548, 2010.