Appendix S4: Analysis of pollen data

S4.1 Fitting the models to the pollen dataset

This appendix is broken into five sections. In the first section, we load and pre-process the pollen data. In the next three sections, we fit the probabilistic models of MVGP, GAM, and BUMMER to the pollen 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 pollen data set.

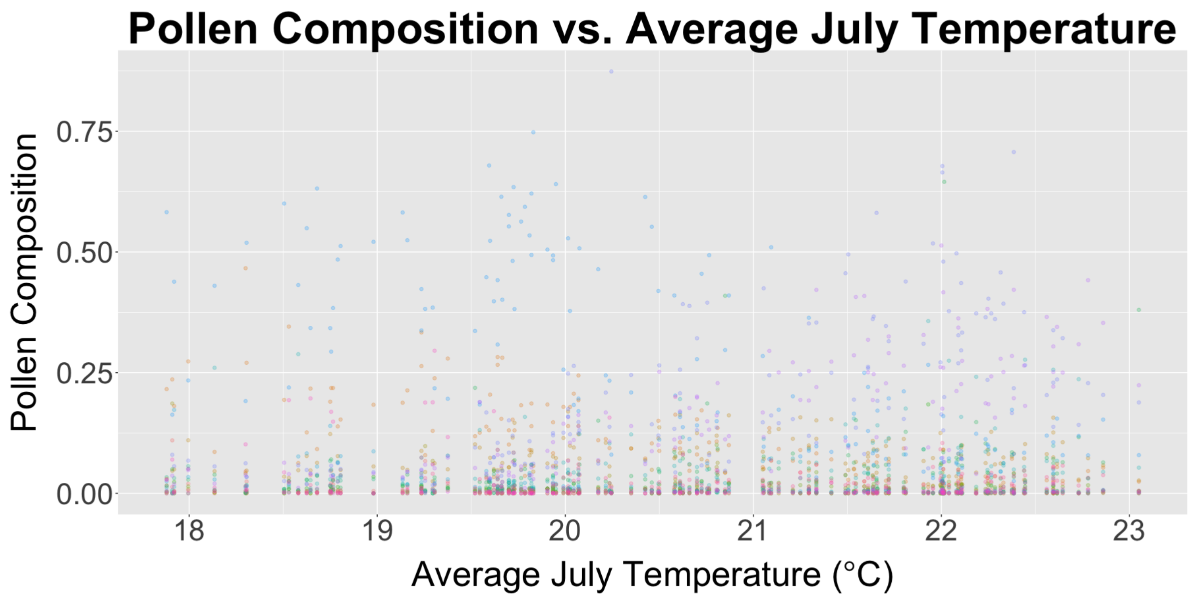
To evaluate the performance of the proposed inverse prediction framework, we fit the proposed models to the pollen dataset consisting of 152 observations of a composition of 16 species along with measurements of average July temperature. First, we load the necessary `R` packages and setting up the file directories for saving model output and progress files.

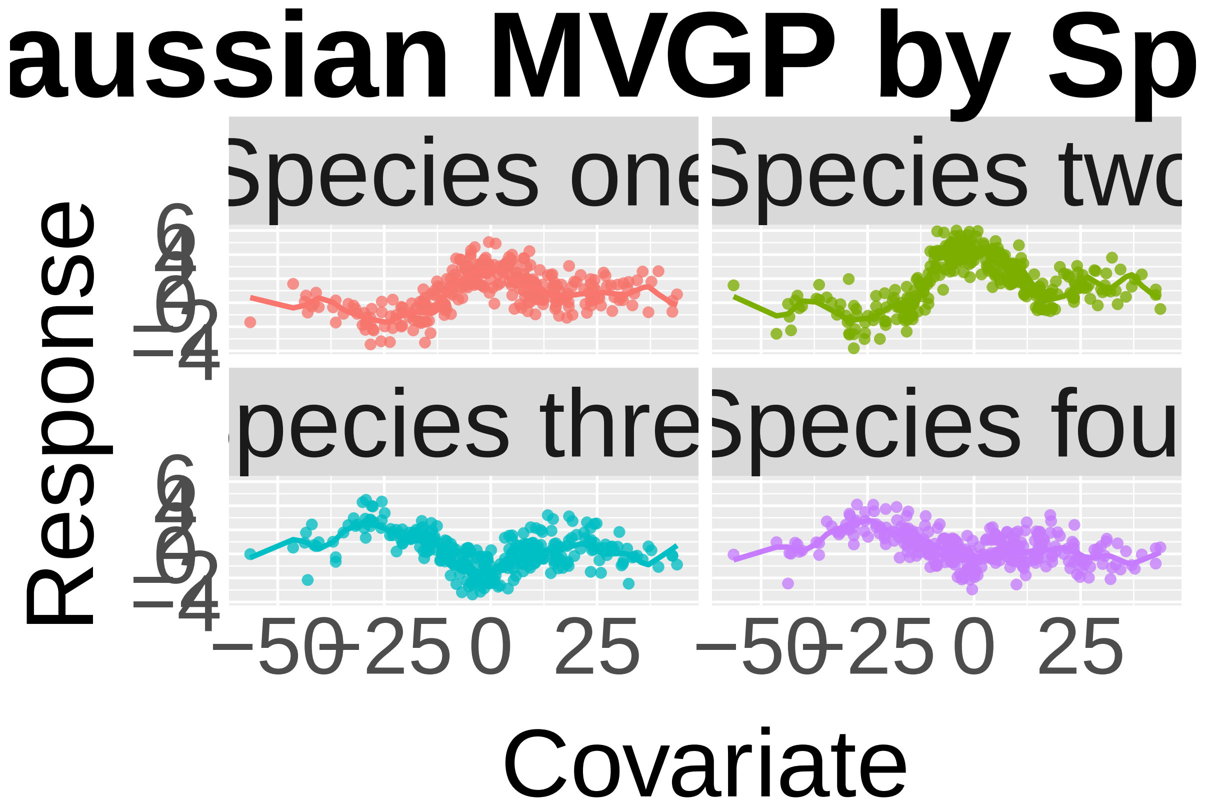
```r
library(BayesComposition)
library(here)
library(ggplot2)
library(GGally)
library(xtable)
library(kableExtra)
library(reshape2)
library(snowfall)
library(parallel)
library(rlecuyer)
library(plyr)
library(knitr)

## 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
```

S4.2 Load pollen data

Next, we load the pollen data. The pollen data consists of 152 observations of 16 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 pollen dataset.

```r
## load and process 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]
colnames(y)  <- revalue(colnames(y),
                        c("Other"                 = "OTHER",
                          "Sum.Other.Deciduous"  = "DECIDUOUS",
                          "Sum.Other.Herbaceous" = "OTHER HERBS",
                          "Sum.Prairie.Herbs"    = "PRAIRIE HERBS",
                          "Sum.Other.Conifer"    = "OTHER CONIFER"))
y <- as.matrix(y)

## load the climate data
X_annual <- as.numeric(dat$tmean_annual[-1])
X <- as.numeric(dat$tmean_07[-1])

## 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, ])
}

## center and scale the covariate value
mean_X <- mean(X)
sd_X <- sd(X)
X <- (X - mean_X) / sd_X

## plot the data
pollenPlotData <- data.frame(
  species = as.factor(rep(colnames(y), each=N)),
  count   = c(as.matrix(y_prop)),
  temp    = rep(X * sd_X + mean_X, times=d)
)

png(file = paste0(save_directory, "pollen-plot.png"), width = 18, height = 9,
    units = "in", res = 100)
ggplot(pollenPlotData, aes(x=temp, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +

  ggtitle("Pollen Composition vs. Average July Temperature") +
  labs(
    x=expression("Average July Temperature ("*degree*C*")"),
    y="Pollen 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, "pollen-plot.png"))
```





S4.3 Fitting the MVGP model to the pollen data

For fitting the MVGP model, we defined 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 pollen 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 ε to model overdispersion, although this could be included for other datasets. The MVGP model took 1.93 hours running 4 MCMC chains in parallel on a 2017 iMac with 4.2GHz processor.

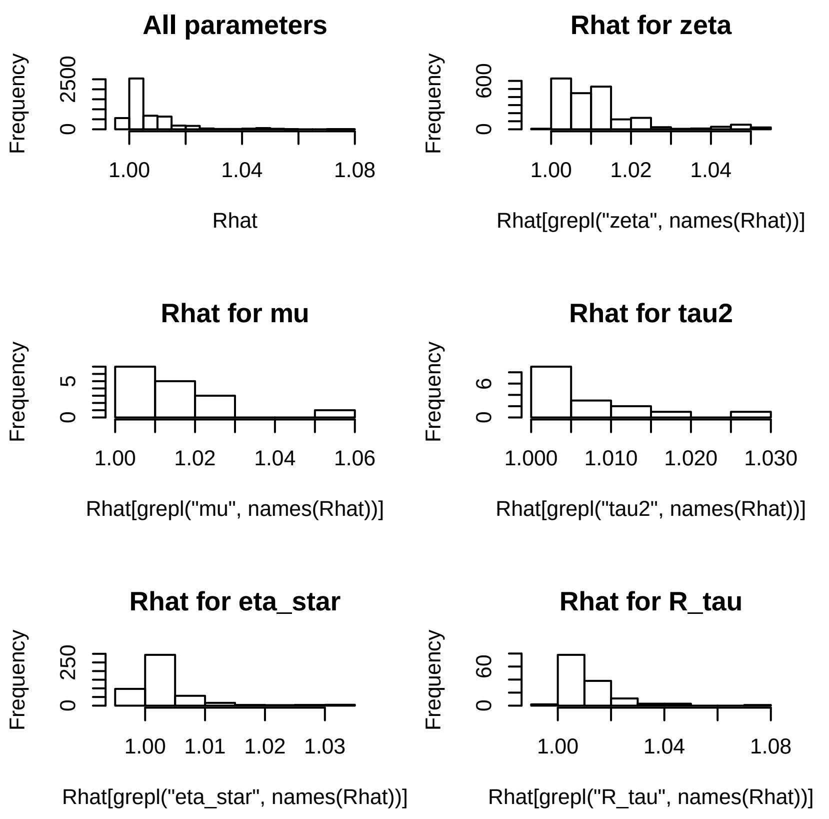
```
n_knots <- 30
X_knots <- seq(min(X, na.rm=TRUE)-1.25*sd(X, na.rm=TRUE),
               max(X, na.rm=TRUE)+1.25*sd(X, na.rm=TRUE), length=n_knots)

## 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,
  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-pollen.RData"))) {
  ## Load MCMC run
  load(paste0(save_directory, "fit-mvgp-pollen.RData"))
} else {
  ##
  ## Long running MCMC
  ##

  ## 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-pollen.txt")

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

S4.3.1 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 $\hat{R} = 1.0756874 \leq 1.1$ at an acceptable value (Gelman et al. [2013], pp.287).

```r
## 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("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] 1.075687
```



S4.3.2 Generate MVGP predictions

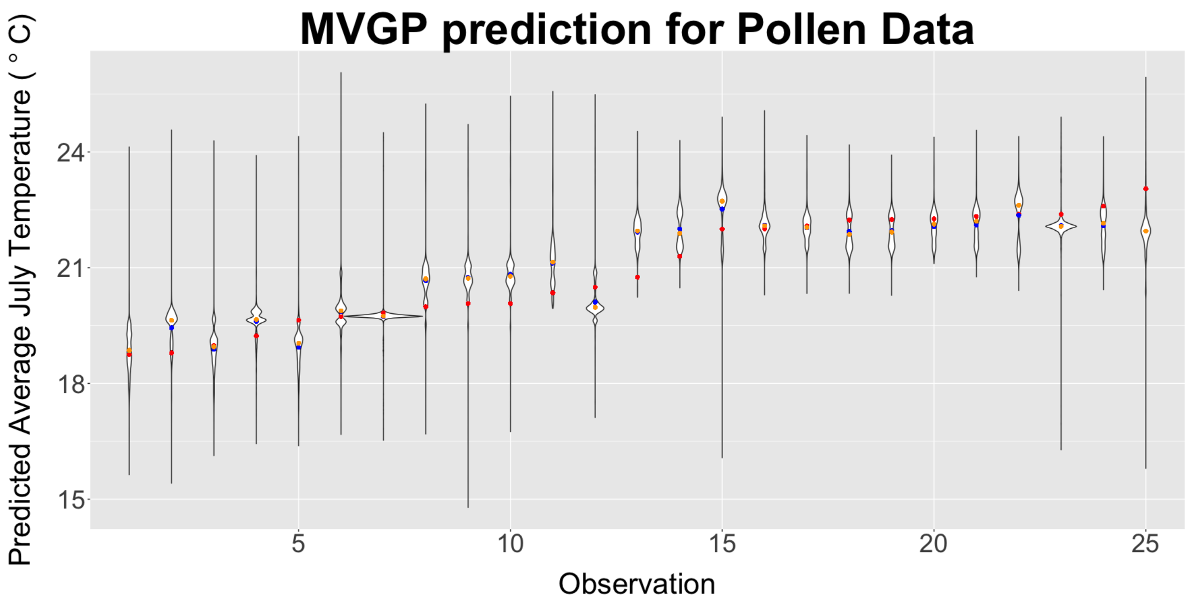
Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the held-out average July temperature using the posterior samples estimated from the calibration model. These estimates can be generated either jointly with the calibration model or estimated using a two-stage 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.

```r
## 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.RData"))) {
  load(paste0(save_directory, "predict-mvgp-pollen.RData"))
} else {
  pred <- predict_compositional_data(
    y[sample_idx, ],
    X[ - sample_idx],
    params,
    samples,
    progress_directory,
    "predict-mvgp-pollen.txt")
  save(pred, file = paste0(save_directory, "predict-mvgp-pollen.RData"))
}
```

S4.3.3 Plot MVGP predictions

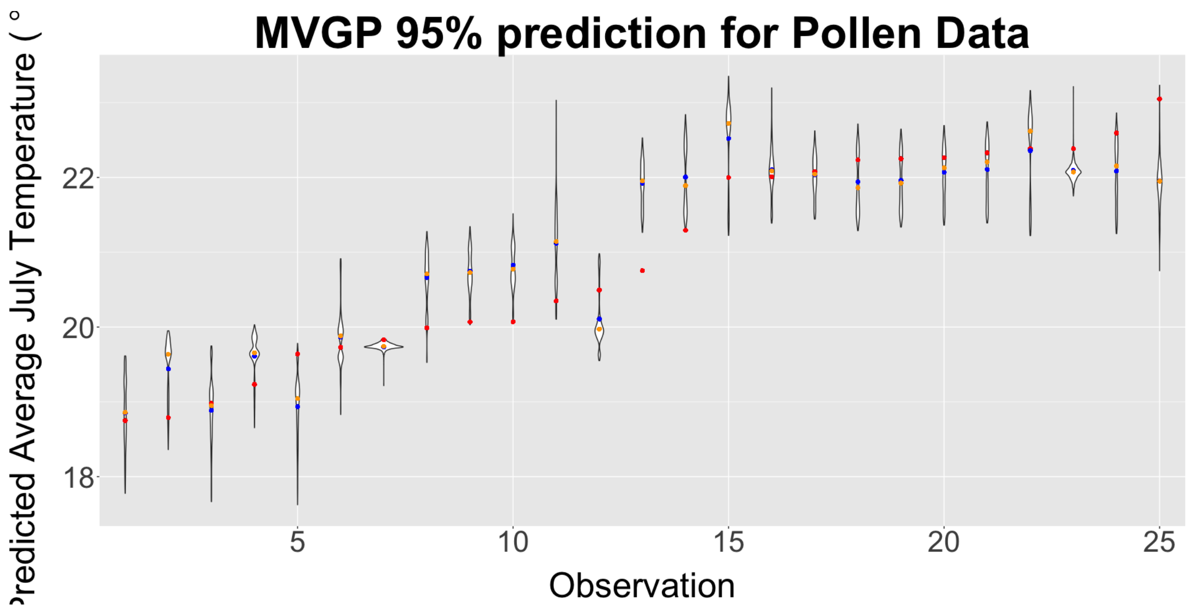
The predictions for 25 randomly chosen hold-out average July temperature observations are shown below. The held-out average July temperature values are shown as red dots and the posterior distribution shown as a violin plot where the width of the violin is proportional to the posterior density. The posterior 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 average July temperature. Notice that the MVGP posterior predictive distribution has larger tails than either the GAM or BUMMER posterior predictive distributions.

```
## 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], 2, median) * sd_X + mean_X,
                    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, "pollen-predictions.png"), width = 18,
    height = 9, units = "in", res = 100)
ggplot(sim_df, aes(observation, covariate)) +
  geom_violin(position="identity", width = 1.85) +
  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=expression("Predicted Average July Temperature ("~degree~C*")")) +
  ggtitle("MVGP prediction for Pollen 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, "pollen-predictions.png"))
```



To better visualize the bulk of the posterior predictive distribution, we display the central 95% of the posterior predictive distribution. The trimmed posterior predictive distributions show more evidence of the multi-modality. For example, observations 6, 10, and 14 show strong multi-modal patterns.

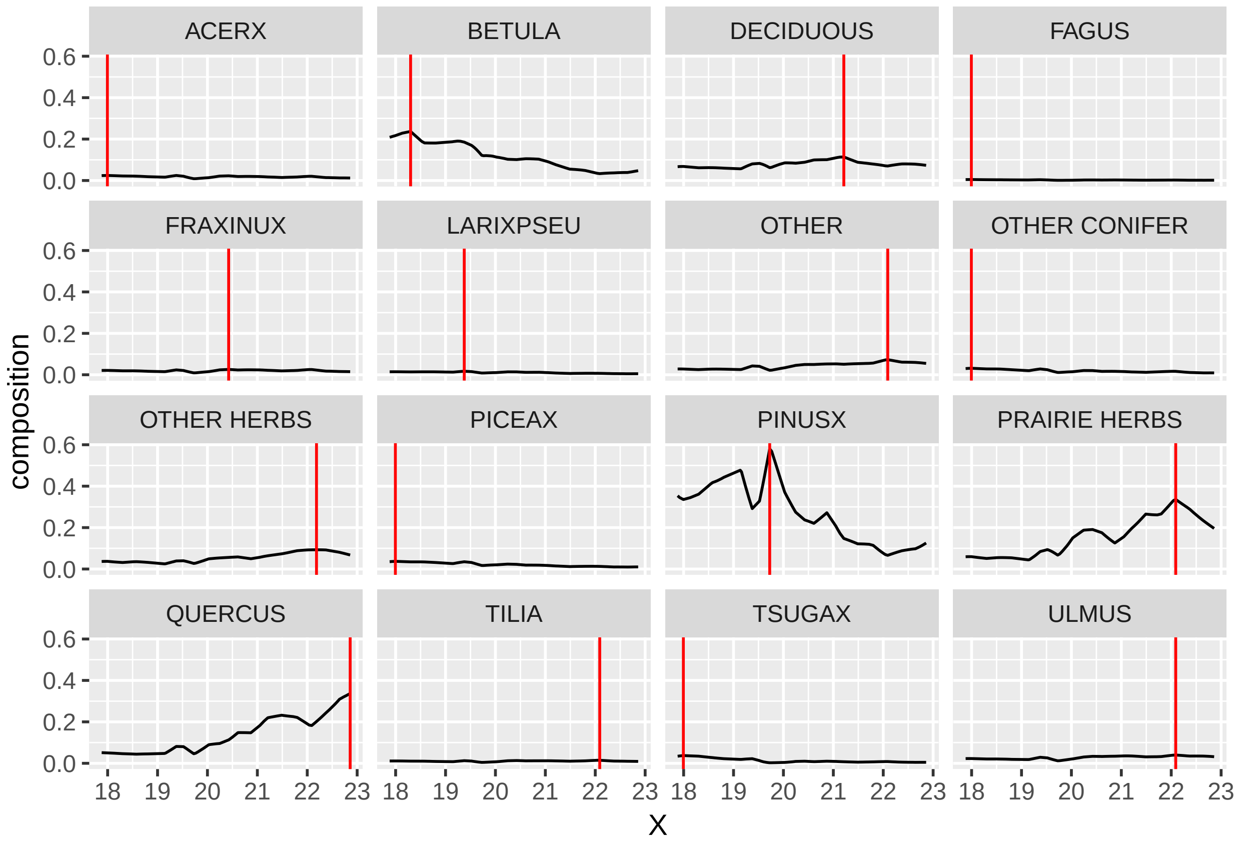
```
X_trimmed <- matrix(0, 0.95*n_samples+1, N_pred)
for (i in 1:N_pred) {
  X_trimmed[, i] <- sort(pred$X[, i])[(n_samples * 0.025):(n_samples * 0.975)]
}
sim_df <- data.frame(
  covariate   = c(X_trimmed[, idx] * sd_X + mean_X),
  mean        = rep(apply(pred$X[, idx] * sd_X + mean_X, 2, mean),
                    each = 0.95*n_samples+1),
  median      = rep(apply(pred$X[, idx], 2, median) * sd_X + mean_X,
                    each = 0.95*n_samples+1),
  observation = factor(rep(1:n_plot, each = 0.95*n_samples+1)),
  truth       = rep(X[sample_idx][idx] * sd_X + mean_X,
                    each = 0.95*n_samples+1))

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

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=expression("Predicted Average July Temperature ("~degree~C*")")) +
  ggtitle("MVGP 95% prediction for Pollen Data") +
  theme(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, "pollen-predictions-trimmed.png"))
```



S4.3.4 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")
```

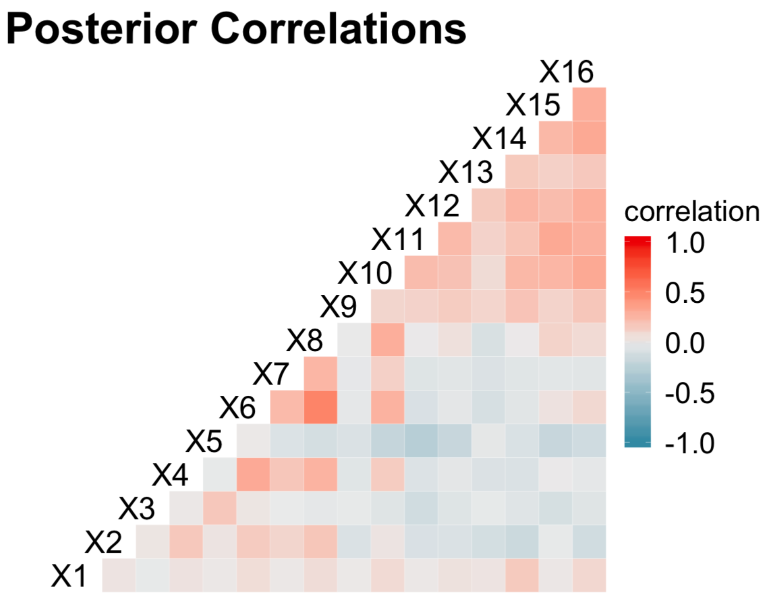


```
order_species <- order(species_maxima, decreasing = FALSE)
```

S4.3.5 Plot MVGP estimated functional correlations

Using the ordering of the species, we plot the estimated correlations in functional response. Notice that species which have similar maximal responses to the covariate tend to have positive functional correlations (correlations near the diagonal are on average more positive that correlations off the diagonal). These functional correlations provide a data-driven method for generating plant 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, , ]
}
rownames(Omega_post_mean) <- colnames(y)

png(file = paste0(save_directory, "pollen-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=6, 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, "pollen-correlations.png"))
```

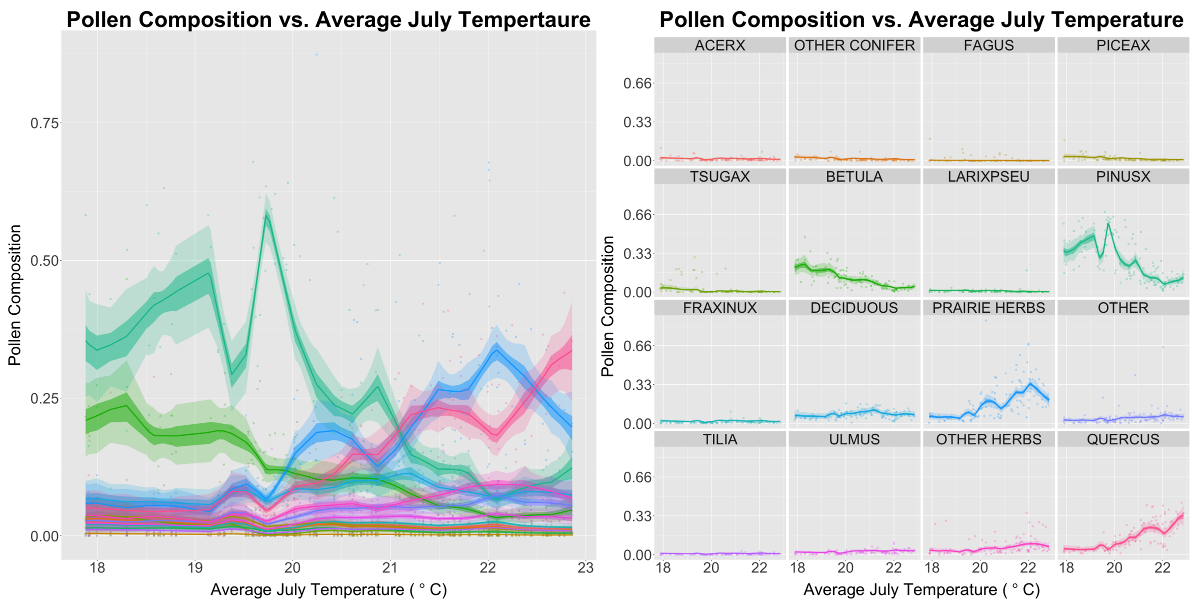


S4.3.6 Plot MVGP estimated functional responses

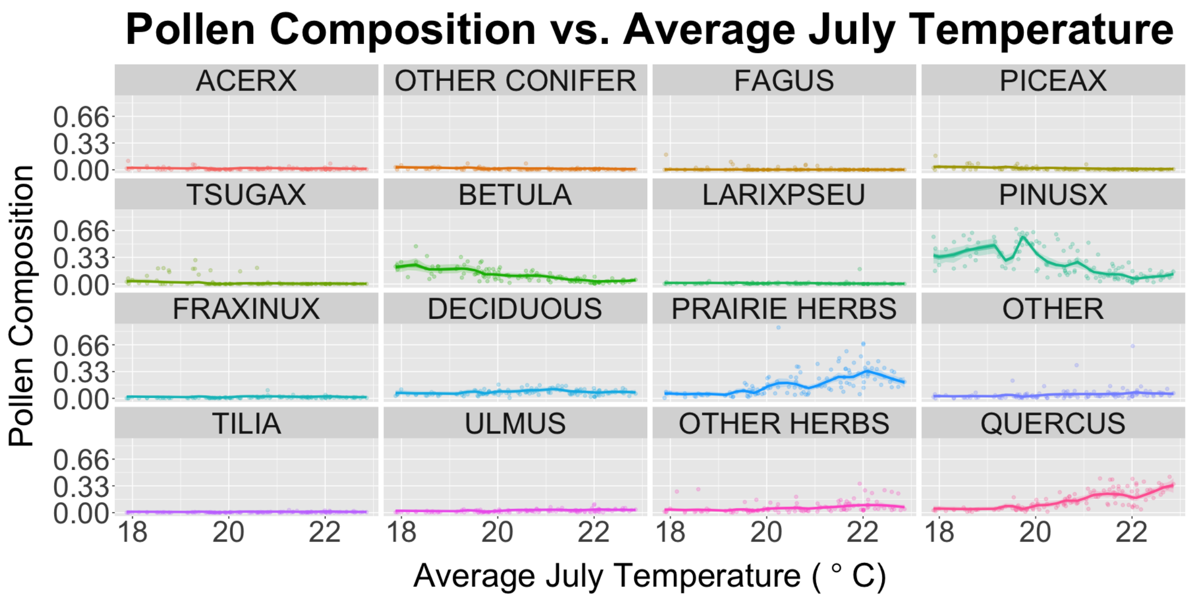
We also plot the estimated mean functional response with associated Bayesian credible intervals (50% central credible interval is darkly shaded, the 95% central credible interval is lightly shaded). These intervals show some surprising results, particularly the behavior of the functional response for *Pinus* which shows a multi-modal functional response. This is due to the *Pinus* genus being composed of multiple species which each have different functional responses to average July temperature.

```
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, ]))
}
```

```r
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, ]),
  temp           = 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=temp, 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("Pollen Composition vs. Average July Tempertaure") +
  theme(legend.position="none",
        plot.title=element_text(size=48, face="bold", hjust=0.5),
        axis.text.x = element_text(size = 32),
        strip.text.x = element_text(size = 26),
        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=temp, y=alpha, col = species), fitPlotData, lwd=1.25) +
  labs(x=expression("Average July Temperature ("~degree~C*")"), y="Pollen Composition")

g2_post <- ggplot(fitPlotData, aes(x=temp, 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("Pollen Composition vs. Average July Temperature") +
  geom_line(aes(x=temp, y=alpha, col = species), fitPlotData, lwd=1.25) +
  facet_wrap( ~ species, ncol = 4) +
  labs(x=expression("Average July Temperature ("~degree~C*")"), y="Pollen Composition") +
  theme(legend.position="none",
        plot.title=element_text(size=48, face="bold", hjust=0.5,
                                margin = margin(t = 0, r = 0, b = 20, l = 0)),
        axis.text.x = element_text(size = 32),
        strip.text.x = element_text(size = 32),
        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)
        )
  ) +
  scale_y_continuous(breaks=c(0.0, 0.33, 0.66, 1.0)) +
  scale_x_continuous(breaks=c(18, 20, 22, 24))

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

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



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

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



S4.4 BUMMER model fit to the pollen 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 50,000 iterations with adaptive Metropolis-Hastings proposals as burn in and thinning every 150 iterations. The model is run for 4 chains resulting in 4,000 posterior samples.

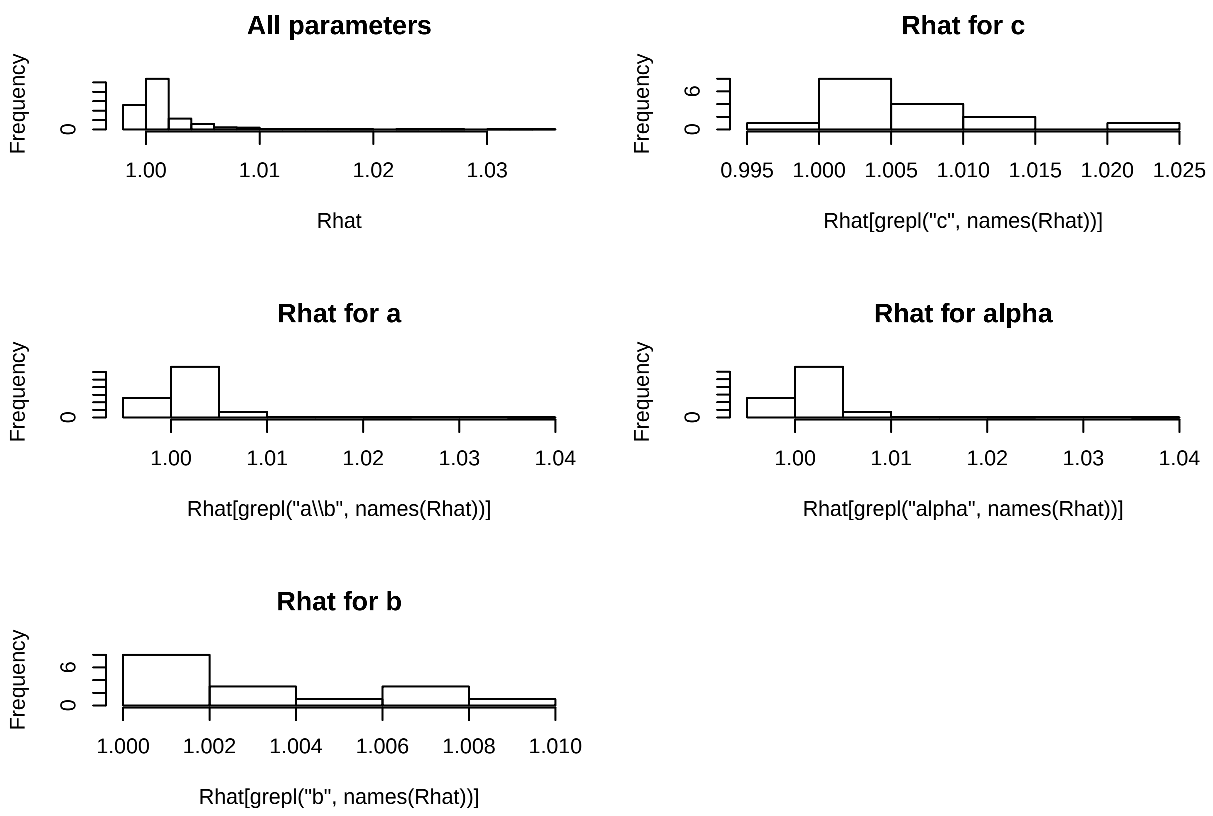
```r
## define the model parameters
params <- list(
  n_adapt                    = 50000,
  n_mcmc                     = 150000,
  n_thin                     = 150,
  likelihood                 = "dirichlet-multinomial",
  function_type              = "bummer",
  n_chains                   = 4,
  n_cores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

if (file.exists(paste0(save_directory, "fit-bummer-pollen.RData"))) {
  ## Load MCMC run
  load(paste0(save_directory, "fit-bummer-pollen.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-pollen.txt")
  save(out, sample_idx, file = paste0(save_directory, "fit-bummer-pollen.RData"))
}
```

S4.4.1 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 $\hat{R} = 1.0350344 \leq 1.1$ 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(mu_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] 1.035034
```



S4.4.2 Generate BUMMER predictions

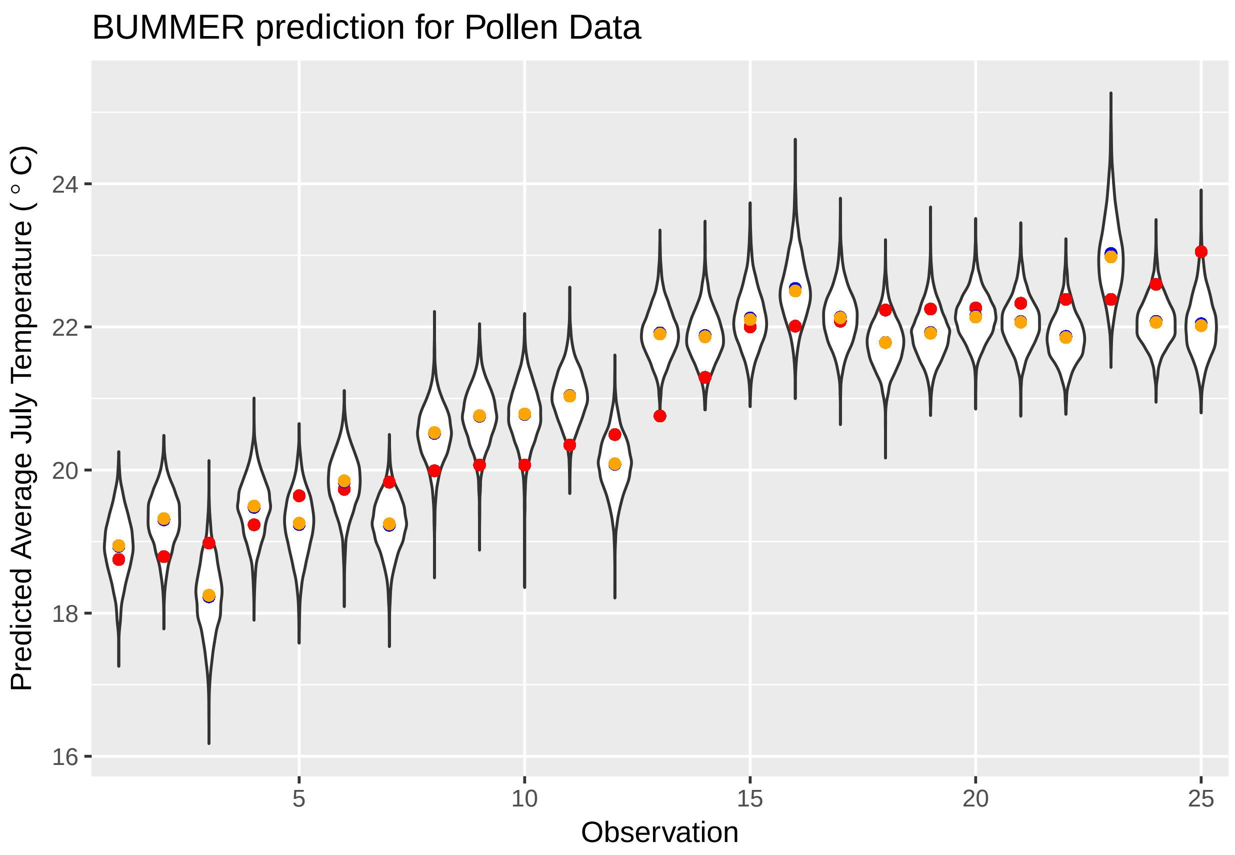
Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved covariates using the posterior samples estimated from the calibration model. These estimates can be generated either jointly with the calibration model or estimated using a two-stage approach 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.

```r
## 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.RData"))) {
  load(paste0(save_directory, "predict-bummer-pollen.RData"))
} else {
  pred <- predict_compositional_data(
    y[sample_idx, ],
    X[ - sample_idx],
    params,
    samples,
    progress_directory,
    "predict-bummer-pollen.txt")
  save(pred, file = paste0(save_directory, "predict-bummer-pollen.RData"))
}
```

S4.4.3 Plot BUMMER predictions

The predictions for the 25 randomly chosen hold-out average July temperature observations are shown below. The held-out temperature 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.

```r
## 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=expression("Predicted Average July Temperature ("~degree~C*")")) +
  ggtitle("BUMMER prediction for Pollen Data")
```



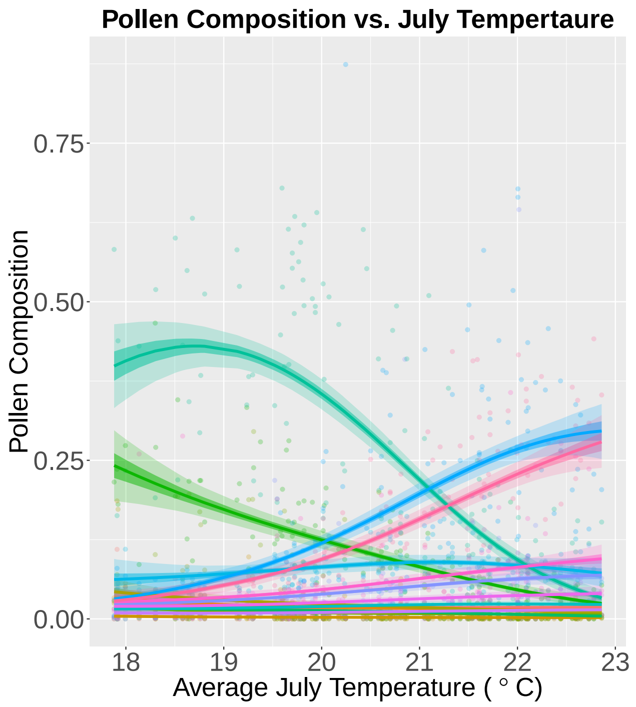
S4.4.4 BUMMER estimated functional response

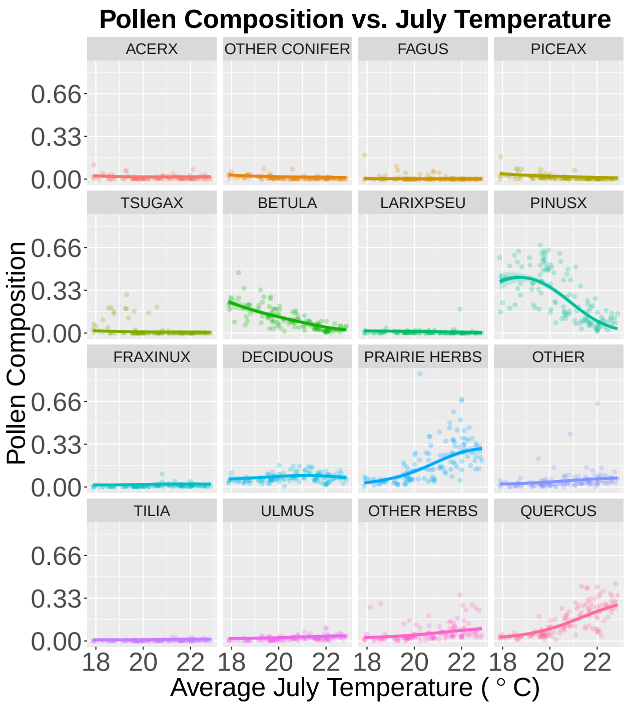
We also plot the estimated mean functional response with associated Bayesian credible intervals (50% central credible interval is darkly shaded, the 95% central credible interval is lightly shaded). We see that the BUMMER model appears to fit the data well; however it violates the known ecology where, for example, the *Pinus* operational taxonomic unit is known to contain multiple species that each have different ecological niches.

```
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, ])
}
```

```r
fitPlotData <- data.frame(
  species        = factor(rep(colnames(y), each=N-N_pred),
                          levels=colnames(y)[order_species]),
  count          = c(y_prop[-sample_idx, ]),
  temp           = 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=temp, 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("Pollen Composition vs. July Tempertaure") +
  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_line(aes(x=temp, y=alpha, col = species), fitPlotData, lwd=1.25) +
  labs(x=expression("Average July Temperature ("~degree~C*")"), y="Pollen Composition", size=20)

g2_post <- ggplot(fitPlotData, aes(x=temp, 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("Pollen Composition vs. July Temperature") +
  geom_line(aes(x=temp, y=alpha, col = species), fitPlotData, lwd=1.25) +
  facet_wrap( ~ species, ncol = 4) +
  labs(x=expression("Average July Temperature ("~degree~C*")"), y="Pollen Composition", size=20) +
  theme(legend.position="none",
        plot.title=element_text(size=24, face="bold", hjust=0.5),
        strip.text.x = element_text(size = 14),
        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)) +
  scale_x_continuous(breaks=c(18, 20, 22, 24))

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

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





S4.5 GAM model fit to the pollen data

The GAM model is a simplification to the full MVGP model. The `BayesComposition` package has an option for fitting the GAM model. We fit the model for 200,000 MCMC iterations, discarding the first 50,000 iterations with adaptive Metropolis-Hastings proposals as burn in and thinning every 150 iterations. The model is run for 4 chains resulting in 4,000 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 β using a log-link function.

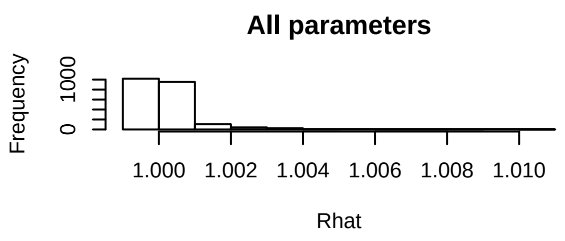
```
## define the model parameters
params <- list(
  n_adapt                    = 50000,
  n_mcmc                     = 150000,
  n_thin                     = 150,
  likelihood                 = "dirichlet-multinomial",
  function_type              = "basis",
  n_chains                   = 4,
  n_cores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

if (file.exists(paste0(save_directory, "fit-gam-pollen.RData"))) {
  ## Load MCMC run
  load(paste0(save_directory, "fit-gam-pollen.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-pollen.txt")

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

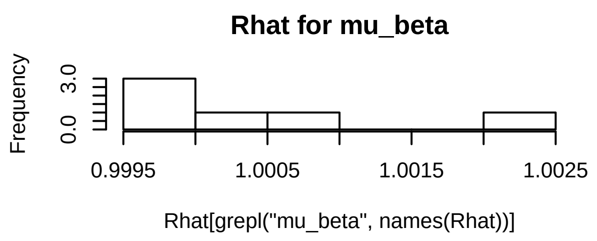
S4.5.1 GAM convergence diagnostics

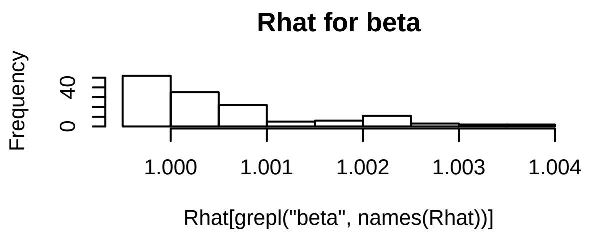
After fitting the model, the posterior samples are generated and MCMC convergence is assessed using the Gelman-Rubin $\hat{R}$ statistic. We see the distribution of $\hat{R}$ statistics is close to 1 with the largest $\hat{R} = 1.0108708 \leq 1.1$ at an acceptable value (Gelman et al. [2013], pp.287).

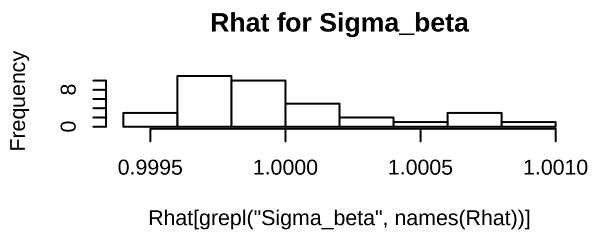
```
## extract posterior samples
samples <- extract_compositional_samples(out)
beta_post <- samples$beta
mu_beta_post <- samples$mu_beta
Sigma_beta_post <- samples$Sigma_beta
alpha_post <- samples$alpha

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)))
```

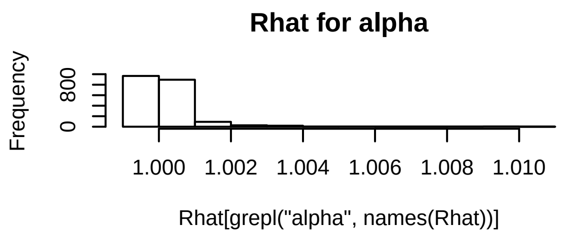
```
## [1] 1.010871
```











S4.5.2 Generate GAM predictions

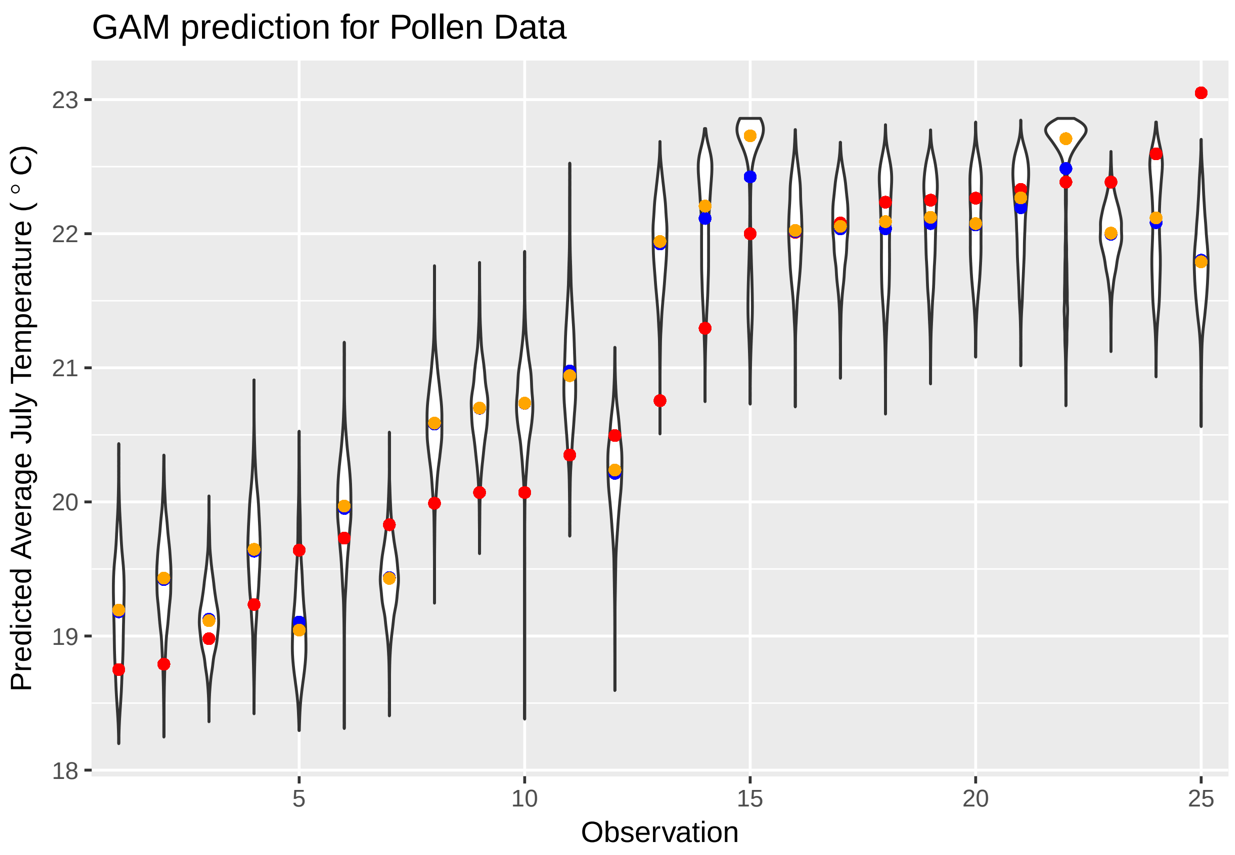
Using the elliptical slice sampler [Murray et al., 2010], we generate posterior predictions for the unobserved covariates using the posterior samples estimated from the calibration model. These estimates can be generated either jointly with the calibration model or estimated using a two-stage approach 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.

```r
## 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.RData"))) {
  load(paste0(save_directory, "predict-gam-pollen.RData"))
} else {
  pred <- predict_compositional_data(
    y[sample_idx, ],
    X[ - sample_idx],
    params,
    samples,
    progress_directory,
    "predict-mvgp-pollen.txt")
  save(pred, file = paste0(save_directory, "predict-gam-pollen.RData"))
}
```

S4.5.3 Plot GAM predictions

The predictions for 25 randomly chosen hold-out average July temperature 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.

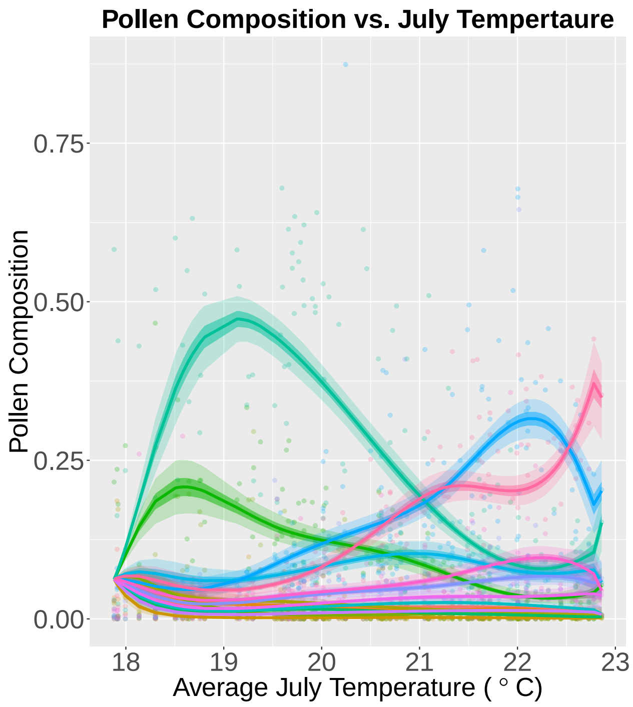
```r
## sorted to increasing values for ease of display
idx <- order(X[sample_idx])
## randomly choose 25 observations to display

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=expression("Predicted Average July Temperature ("~degree~C*")")) +
  ggtitle("GAM prediction for Pollen Data")
```

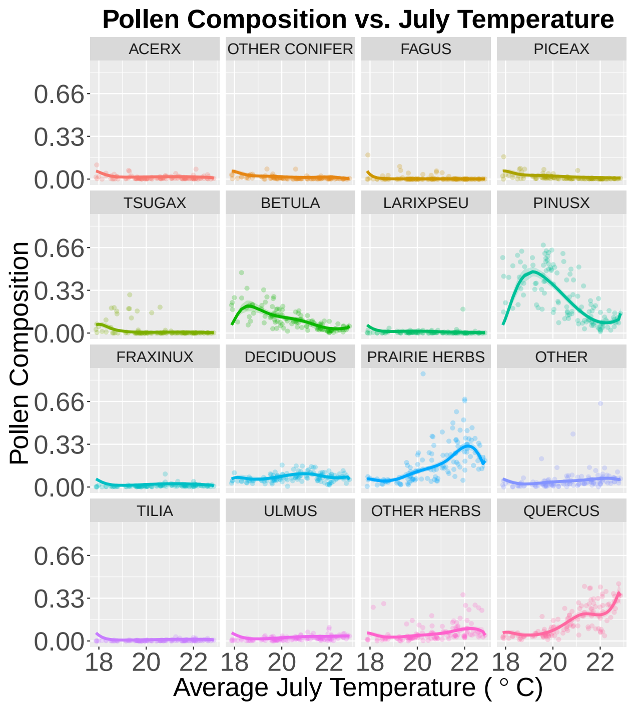


S4.5.4 GAM estimated functional response

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).

```r
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, ]),
  temp           = 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=temp, 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("Pollen Composition vs. July Tempertaure") +
  theme(legend.position="none",
        plot.title=element_text(size=24, face="bold", hjust=0.5),
        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=temp, y=alpha, col = species), fitPlotData, lwd=1.25) +
  labs(x=expression("Average July Temperature ("~degree~C*")"), y="Pollen Composition", size=20)

g2_post <- ggplot(fitPlotData, aes(x=temp, y=count, color=species, group=species)) +
  geom_point(alpha=0.25) +
  theme(legend.position="none") +
  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("Pollen Composition vs. July Temperature") +
  geom_line(aes(x=temp, y=alpha, col = species), fitPlotData, lwd=1.25) +
  facet_wrap( ~ species, ncol = 4) +
  labs(x=expression("Average July Temperature ("~degree~C*")"), y="Pollen Composition", size=20) +
  theme(legend.position="none",
        plot.title=element_text(size=24, face="bold", hjust=0.5),
        strip.text.x = element_text(size = 14),
        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)) +
  scale_x_continuous(breaks=c(18, 20, 22, 24))

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





S4.6 Cross-validation for the pollen data

Finally, we provide code for the 12-fold cross-validation experiment. Each of the probabilistic models was fit to the training pollen 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 4.1 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)
```

```r
## 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)
```

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

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

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

```r
sfLibrary(BayesComposition)
```

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

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

```r
# export global variables


## Fit cross-validation
for (model_name in models) {
  if (!file.exists(paste0(save_directory, "cv-pollen-", model_name, ".RData"))) {
    # if (!file.exists(here::here("model-fit", paste0("dm-mvgp-cv-pollen-",
    #                                                  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, "-pollen.txt"))

      sink(paste0(progress_directory, "cv-", model_name, "-pollen.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, "-pollen.txt"))

      sink(paste0(progress_directory, "cv-", model_name, "-pollen.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, "-pollen.txt"))

      sink(paste0(progress_directory, "cv-", model_name, "-pollen.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, "-pollen.txt"),
      dataset            = "pollen"
    )

    if (model_name == "MVGP" || model_name == "BUMMER" || model_name == "GAM") {
      ## end timing
      sink(paste0(progress_directory, "cv-", model_name, "-pollen.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,
```

```r
        file = paste0(save_directory, "cv-pollen-", model_name, ".RData"))
  }
}
## ends snowfall session
sfStop()
```

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

S4.6.1 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 pollen data in cross-validation, the MAT generates the best point predictions based on MSPE and MAE followed closely by MVGP, GAM, and WA. MVGP generates the best probabilistic predictions based on CRPS with the other Bayesian models performing well on this metric. The Bayesian predictions based on the pollen data are better calibrated than those using the pollen data based on credible interval coverage.

```r
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-pollen-", 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)))
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-pollen.tex"),
#       floating = FALSE)

kable(cv_results, digits = 4)
```

	CRPS	MSPE	MAE	95% CI coverage
MVGP	0.2746	0.2178	0.3864	87.5000
GAM	0.2828	0.2119	0.3912	78.9474
BUMMER	0.2900	0.2489	0.4009	80.2632
WA	0.3711	0.2287	0.3711	94.0789
MAT	0.3486	0.1821	0.3486	100.0000
MLRC	0.3873	0.2722	0.3873	96.7105

References

Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2013.

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

Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009.