tolerance curves plotting

Silas Tittes

18 October, 2017

```
knitr::opts_knit$set(
  root.dir = '~/Documents/Projects/tolerance-curve2/'
  )
```

sim_params3 ## Data prep

```
source("load_data.R")
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## col_factor(): readr, scales
## discard():    purrr, scales
## filter():     dplyr, stats
## lag():        dplyr, stats

##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract

## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.

##

## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.

##
## Attaching package: 'gdata'

## The following objects are masked from 'package:dplyr':
##
##     combine, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     keep

## The following object is masked from 'package:stats':
##
##     nobs

## The following object is masked from 'package:utils':
##
##     object.size

## The following object is masked from 'package:base':
##
##     startsWith

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':
##
##     expand

## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded glmnet 2.0-13

## Loading required package: rgl

## Loading required package: maps

##
## Attaching package: 'maps'

## The following object is masked _by_ '.GlobalEnv':
##
##     map

## The following object is masked from 'package:purrr':
##
##     map
```

```
##
## Attaching package: 'phytools'

## The following object is masked from 'package:Matrix':
##
##     expm

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## Loading required package: StanHeaders

## rstan (Version 2.15.1, packaged: 2017-04-19 05:03:57 UTC, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## rstan_options(auto_write = TRUE)
## options(mc.cores = parallel::detectCores())

##
## Attaching package: 'rstan'

## The following object is masked from 'package:magrittr':
##
##     extract

## The following object is masked from 'package:tidyr':
##
##     extract

##
## Setting initial dates...
## Fitting in progress... get a first set of estimates
##          Penalised log-lik = -1.881602
## Optimising rates... dates... -1.881602
##
## Done.
```

```
draws <- read_csv("bayes/stan_par1_df.csv") #draws for penalized zero model
```

```
## Parsed with column specification:
## cols(
##   Species = col_character(),
##   d = col_double(),
##   draw = col_integer(),
##   e = col_double(),
##   a = col_double(),
##   b = col_double(),
```

```
##   c = col_double(),
##   beta_0 = col_double(),
##   beta_1 = col_double(),
##   nu = col_double(),
##   maxima = col_double(),
##   breadth = col_double(),
##   area = col_double(),
##   special = col_double()
## )
```

```
draw_fits <- read_csv("bayes/fitted_points_mod1.csv")
```

```
## Parsed with column specification:
## cols(
##   Species = col_character(),
##   x = col_double(),
##   y = col_double(),
##   draw = col_integer()
## )
```

Trace plots and stan summary

```
#note parameters as discussed in paper are:
  #a = alpha
  #b = beta
  #c = zeta
  #d = delta
  #e1 = epsilon
#traceplot for parameter for each species


stan_sum <- summary(stanDat, par = c("a", "b", "c", "d", "e",
                                     "beta_0", "beta_1", "nu"))$summary


options(xtable.sanitize.colnames.function=identity,
          xtable.sanitize.rownames.function=identity)
print.xtable(
  xtable(round(stan_sum, 2)), comment = F, file = "stan_table.tex",
  tabular.environment='longtable', include.colnames = TRUE,
  floating=FALSE, add.to.row = list(pos = list(0), command = "\\hline \\endhead "))

pdf("figures/trace_a.pdf")
rstan::traceplot(stanDat, par=c("a"))
dev.off()
```

```
## pdf
##   2
```

```
pdf("figures/trace_b.pdf")
rstan::traceplot(stanDat, par=c("b"))
dev.off()
```

```
## pdf
##   2
```

```
pdf("figures/trace_c.pdf")
rstan::traceplot(stanDat, par=c("c"))
dev.off()
```

```
## pdf
##   2
```

```
pdf("figures/trace_d.pdf")
rstan::traceplot(stanDat, par=c("d"))
dev.off()
```

```
## pdf
##   2
```

```
pdf("figures/trace_e.pdf")
rstan::traceplot(stanDat, par=c("e"))
dev.off()
```

```
## pdf
##   2
```

```
pdf("figures/trace_beta_0.pdf")
rstan::traceplot(stanDat, par=c("beta_0"))
dev.off()
```

```
## pdf
##   2
```

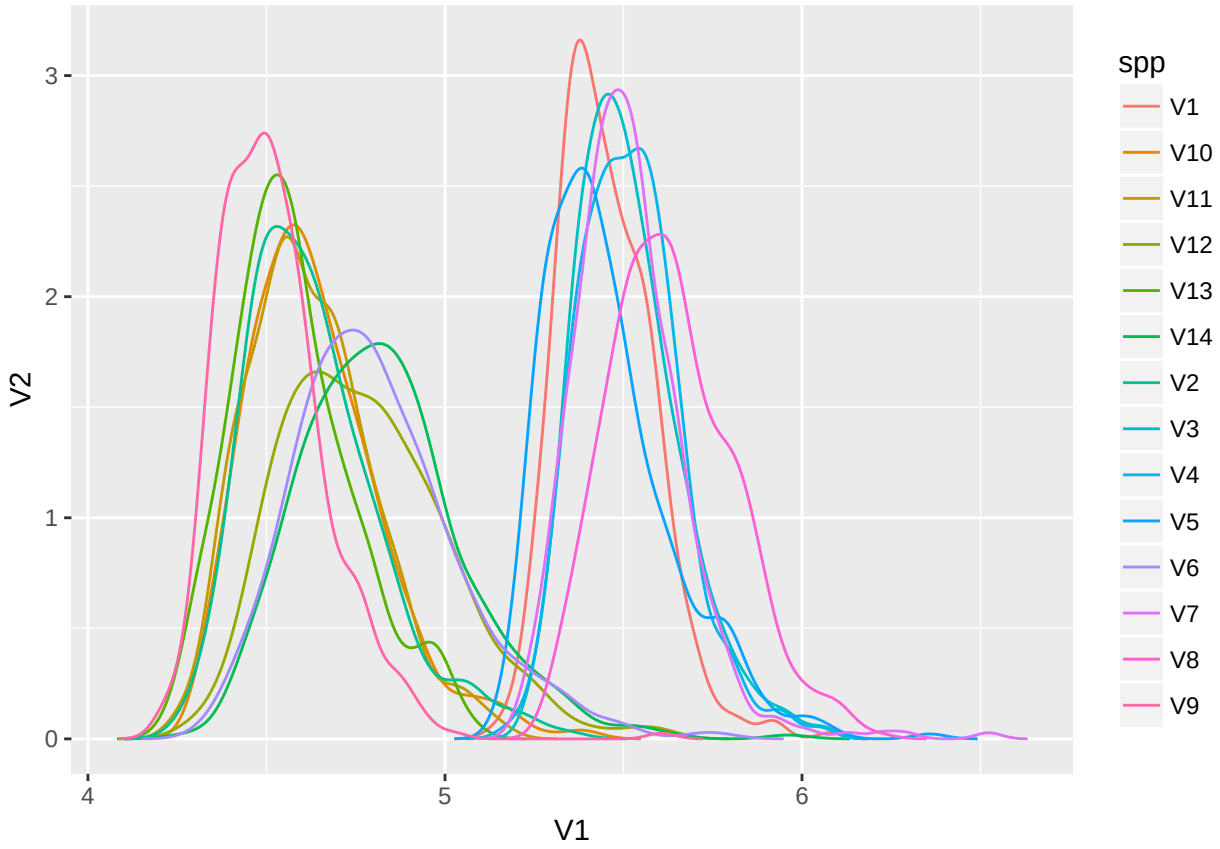
```
pdf("figures/trace_beta_1.pdf")
rstan::traceplot(stanDat, par=c("beta_1"))
dev.off()
```

```
## pdf
##   2
```

```
pdf("figures/trace_nu.pdf")
rstan::traceplot(stanDat, par=c("nu"))
dev.off()
```

```
## pdf
##   2
```

```
posts$e %>%
  as_tibble %>%
  gather("spp", "param") %>%
  group_by(spp) %>%
  do({
    den1 <- density(.$param)
    return(as_tibble(cbind(den1$x, den1$y)))
  }) %>% ggplot(aes(x = V1, y = V2, colour = spp)) +
  geom_line()
```




```
  #geom_line(
  #  data = data.frame(
  #    x = seq(0,20),
  #    y = dgamma(seq(0,20), shape = 2, rate = .2)
  #    ),
  #  aes(x=x,y=y),
  #  colour="red", lty = 2)
```

Raw parameter prob differences

```
comp_matrices <- function(var, cut = 0.95){

  #high <- 1 - (1 - cut)/2
```

```
  #low <- (1 - cut)/2
  high <- cut
  low <- 1 - cut

  #var <- post_params[1]
  var_df <- draws %>%
    select(Species, draw, one_of(var)) %>%
    rename_(var = var) %>%
    spread(key = Species, value = var) %>%
    select(-draw)

  map_comp_p <- map_df(var_df, ~{
    map_dbl(var_df, function(x){
      comp_p <- mean(.x > x)
      ifelse(comp_p > high | comp_p < low, round(mean(.x - x),2), NA)
    })
  }) %>% as.matrix

  map_comp_p[upper.tri(map_comp_p, diag = T)] <- NA
  map_comp_p <- cbind(colnames(map_comp_p), map_comp_p)
  colnames(map_comp_p)[1] <- tab_strings[which(post_params == var)]

  write.csv(map_comp_p,
            file = paste0("analyses_and_viz/parameter_comp_matrices/", var,"_matrix.csv"))

  options(xtable.sanitize.colnames.function=identity,
          xtable.sanitize.rownames.function=identity)

  print.xtable(xtable(map_comp_p), size="\\fontsize{5pt}{5pt}\\selectfont",
               file = pair_file, append = TRUE, include.rownames=FALSE)

}


post_params <- c("c", "d", "e", "maxima", "breadth", "area" ,"special")
tab_strings <- c("$\\zeta$", "$\\delta$", "$\\epsilon$", "maxima", "$\\epsilon - \\delta$", "area", "$\\
names(post_params) <- tab_strings
pair_file <- "analyses_and_viz/parameter_comp_matrices/post_all_comparisons.tex"
file.create(pair_file)
```

```
## [1] TRUE
```

```
post_params %>% map( ~ comp_matrices(var = .x)) %>% invisible
```

Plotting phylogenetic signal of curves

```
sig <- read_csv("derived_files/curve_K.csv")
```

```
## Parsed with column specification:
## cols(
```

```
##   signal = col_double(),
##   param = col_character(),
##   tree = col_integer()
## )
```

```
sig %>% group_by(param) %>%
  summarise(quant = quantile(signal, probs = 0.975)) %>%
  arrange(quant)
```

```
## # A tibble: 8 x 2
##     param     quant
##     <chr>     <dbl>
## 1   curve 0.1495462
## 2  maxima 0.1865242
## 3       d 0.2880789
## 4 special 0.3614579
## 5    area 0.3750004
## 6       e 0.3878204
## 7       c 0.5672559
## 8 breadth 0.5697712
```

```
#sig_den <- sig %>% group_by(param) %>%
#  do(den = density(sig$signal))
library(stringr)

cairo_pdf("~/Desktop/signal_draft.pdf")
sig %>%
  mutate(param = str_replace_all(sig$param,
                c("^c$" = " ",
                  "^d$" = " ",
                  "^e$" = " ")
                )) %>%
  ggplot(aes(x = signal)) +
  geom_density(fill="blue", colour = "white", alpha = 0.2) +
  facet_wrap(~param, nrow = 2) +
  theme_bw() +
  theme(text = element_text(size=16))
dev.off()
```

```
## pdf
##   2
```

Parameter density plot with phylogeny version 2

```
plot_concept <- function(){
  #conceptual plot over top of densities
  off <- 0.2
  a <- 4
  b <- 2.7
  c <- 5.5
```

```
  d <- 0
  e <- 1
  xseq_0 <- seq(0,1, length.out = 500)
  xseq_a <- xseq_0 * (e - d) + d
  fa = stretch_kumara(x = xseq_0, a, b, c)
  opt_loc <- (((a - 1)/(a*b - 1))^(1/a) * (e - d) + d)
  low_loc <- d
  high_loc <- e
  bredth <- e - d

  arr_len <- 0.05
  par_cex <- 1.4

  plot(xseq_a, fa, type = "l", lwd = 4, axes = F, xlab = "", ylab = "",
       ylim = c(0, max(fa)*1.25) )
  axis(1, labels = F); axis(2, labels = F)
  mtext(text = "Fitness", side = 2, line = 1.2)
  mtext(text = "Environment", side = 1, line = 1.2)

  #polygon for area
  polygon(x = c(rev(xseq_a), xseq_a), y = c(rep(0, length(fa)), fa),
          density = 100, col = alpha("black", 0.05), border = T, lwd = 2)
  text(0.7, mean(fa), "Area")

  text(x = opt_loc, y = max(fa)*(1-off), "maxima")
  points(x = opt_loc, y = max(fa), pch = 21, bg = "white", cex = 1.2)


  text(x = low_loc, y = max(fa)*off, expression(delta), cex = par_cex)
  points(x = low_loc, y = 0, pch = 21, bg = "white", cex = 1.2)

  text(x = high_loc, y = max(fa)*off, expression(epsilon), cex = par_cex)
  points(x = high_loc, y = 0, pch = 21, bg = "white", cex = 1.2)

  #segment for zeta
  #calc kums with zeta = 1
  z1 <- max(stretch_kumara(x = xseq_0, a, b, 1))
  za <- max(fa)
  zeta_loc <- low_loc + 0.25
  arrows(x0 = zeta_loc, y0 = za, x1 = zeta_loc, y1 = z1,
         code = 3, angle = 90, length = arr_len, lwd = 2, col = "grey50")
  text(x = zeta_loc + 0.02, y = (z1+za)/2, expression(zeta), cex = par_cex)


  yy <- 4
  arrows(x0 = 0, y0 = max(fa)+1, x1 = high_loc, y1 = max(fa)+1,
         code = 3, angle = 90, length = arr_len, lwd = 2, col = "grey50")

  text(x = median(xseq_a), y = max(fa)+2.5, expression(epsilon - delta), cex = par_cex)

}
```

```
source("derived_files/state_reg.R")

#set habitat colors
colz <- topo.colors(n = 12, alpha = 0.7)[c(7,11,4)]
colz2 <- topo.colors(n = 12, alpha = 1)[c(7,11,4)]
names(colz) <- c("aqua_terr", "terrestrial","vernal")
colz_par <- colz[match(state_reg, names(colz))]
names(colz_par) <- names(state_reg)


spp_list <- rev(lasth$tip.label[lasth$edge[,2][lasth$edge[,2] <= length(lasth$tip.label)]])

pdf("figures/fig4_curves.pdf")

cnc <- 1
ml <- rbind(c(0, 0, 0, 0, 0),
            c(0, 1, 1, 1, 0),
            c(0, 1, 1, 1, 0),
            c(0, 1, 1, 1, 0),
            cbind(rep(2,length(spp_list)),
            matrix(seq_along(spp_list)+1+cnc),
            matrix(seq_along(spp_list)+length(spp_list)+1+cnc),
            matrix(seq_along(spp_list)+2*length(spp_list)+1+cnc),
            matrix(seq_along(spp_list)+3*length(spp_list)+1+cnc)),
            rep(0, 5), rep(0, 5))

layout(ml)

par(mar = c(2,2,0,0))
plot_concept()
par(mar = c(0,0,0,0))

plot.phylo(lasth, edge.width = 2, show.tip.label = F)

post_params <- c("d", "maxima", "e", "breadth")
greek_params <- c(expression(delta), "maxima", expression(epsilon), "breadth")


seq_along(post_params) %>% map(function(x){
  var <- post_params[x]
den_var <- draws %>%
  group_by(Species) %>%
  do(den = density(.[[var]]))

den_range <- den_var$den %>% map_df(~{
  xl = quantile(.x$x, 0)
  xh = quantile(.x$x, 1)
  yl = 0
  yh = quantile(.x$y, 1)
  cbind(xl, xh, yl, yh) %>% as_data_frame
}) %>% summarise_each(funs(min, max))
```

```
names(den_var$den) <- den_var$Species
seq_along(spp_list) %>% map(~{
 spp_den <- den_var$den[[spp_list[.x]]]
 plot(spp_den$x, spp_den$y,
      xlim = c(den_range$xl_min, den_range$xh_max),
      ylim = c(0, den_range$yh_max),
      axes = F, type = "n")

 polygon(spp_den$x, spp_den$y, col = colz_par[spp_list[.x]], border  = T)

 if(var == "d") legend("left", spp_list[.x], bty = "n", cex = 0.8)

 })
axis(1)
mtext(greek_params[x], side = 1, line = 2)

}) %>% invisible

dev.off()
```

```
## pdf
##   2
```

curve plots

```
holo <- c("ferrisiae", "glabrata", "coulteri", "chrysantha")
emery_zeros <- read.xls("data/Inundation_compiled_FINAL.xlsx") %>%
  mutate(Inflor_biomass = ifelse(
    is.na(Inflor_biomass) &
      ifEmerge.Y.N. == 1, yes =  0,
    no = Inflor_biomass)) %>%
  mutate(
    treat = ifelse(
      Treatment == "F", yes = 5,
      no = ifelse(
        Treatment == "MF", yes = 4,
        no = ifelse(
          Treatment == "B", yes = 3,
          no = ifelse(
            Treatment == "MD", yes = 2,
            no = 1
          )
        )
      )
    )
  ) %>%
  filter(!is.na(Inflor_biomass)) %>%
  group_by(Species) %>%
  mutate(sppint = as.integer(Species)) %>%
  group_by(Species, treat) %>%
  filter(sum(Inflor_biomass > 0) == 0) %>%
```

```
  ungroup() %>%
  mutate(
    Species_h = ifelse(
      as.character(Species) %in% holo,
      "hologymne", as.character(Species)
    ),
    sppint_h = as.integer(as.factor(Species_h))
  )


mean_draws <- draws %>% group_by(Species) %>%
  summarise(a = mean(a),
            b = mean(b),
            c = mean(c),
            d = mean(d),
            e = mean(e),
            draw = 1)


colz <- topo.colors(n = 12, alpha = 1)[c(7,11,4)]

#load reg_fit -- simmaps
source("derived_files/simmap.R")
source("derived_files/state_reg.R")

sppMaxVal <- by(data = emery$Inflor_biomass,
                INDICES = emery$sppint, FUN = max)
names(sppMaxVal) <- unique(emery$Species)


#from liam revell's blog
simmap_prop <-function(x){
  y<-sapply(x$maps,function(x) names(x)[1])
  names(y)<-x$edge[,1]
  y<-y[as.character(length(x$tip)+1:x$Nnode)]
  return(y)
}


#get colors mapped to habitats for OUwie
states <- colnames(reg_fit[[1]]$mapped.edge)
names(states) <- colz
colz <- names(states)
names(colz) <- states


habz <- colnames(reg_fit[[1]]$mapped.edge)
sim_states <- sapply(reg_fit, simmap_prop)
pies_sim <-t(apply(sim_states,1,
               function(x,levels,Nsim){
                 summary(factor(x,levels))/Nsim},
               levels=habz, Nsim=ncol(sim_states)))
```

```r
#plot data
spp_list <- rev(lasth$tip.label[lasth$edge[,2][lasth$edge[,2] <= length(lasth$tip.label)]])
cut <- 0.99
lo <- (1 - cut)/2
hi <- 1 - (1 - cut)/2
xrng <- draw_fits$x %>% quantile(c(lo, hi))
yrng <- max(emery$Inflor_biomass)


#pdf(paste0(getwd(),"/figures/tree_tolerance.pdf"))
#png(filename = "figures/fig3_tree_tolerance.png") #alt
pdf("figures/fig3_tree_tolerance.pdf") #alt
#svg("figures/fig3_tree_tolerance.svg") #alt

c1 <- rep(1, length(spp_list))
c_2_4 <- rep(0, length(spp_list))
c3 <- 2:(length(spp_list)+1)
m_lay <- cbind(c1, c1, c1, c3, c3, c_2_4)
m_lay <- rbind(rep(0, ncol(m_lay)), m_lay, rep(0, ncol(m_lay)))
layout(m_lay,
       widths = c(0.25, 0.25, 0.25, 0.2, 0.2, 0.05),
       heights = c(0.01, rep(1/(nrow(m_lay)-2), (nrow(m_lay)-2)), 0.05) )

par(mar=c(0,0,0,0))

plot.phylo(lasth, show.tip.label = F, edge.width = 4,
           no.margin = T)
#nodelabels(pie=pies_sim, piecol=colz, cex = 1.2)
tiplabels(pch = 22, cex = 3,
          bg = colz[state_reg[lasth$tip.label]])

#format node support for labeling
node_fmt <- ifelse(lasth$node.label < 1,
                   substring(sprintf("%4.2f", lasth$node.label), 2),
                   lasth$node.label)

nodelabels(node_fmt, frame = "n",
           cex = 1, font = 2, pos=4, offset = 1)


legend("bottomleft", c("Vernal pool / terrestrial", "Terrestrial", "Vernal pool"), pch = 22, pt.cex = 1
       pt.bg = colz, cex = 1.2, bg = "white")


for(spp in spp_list){

  #spp <- spp_list[14]
  #plt_draws <- 500 #!!!!
  n_draws <- posts$lp__ %>% nrow
  spp_draws <- draw_fits %>% filter(draw <= n_draws, Species == spp)

  par(mar=c(0,2,0,0))
```

```
  plot(
    NA, NA,
    ylim = c(0, yrng), xlim = xrng,
    axes=F, xlab="", ylab=""
  )
  box()

  1:n_draws %>% map(~ {
    c_draw <- spp_draws %>% filter(draw == .x)
    lines(c_draw$x, c_draw$y,
          col=alpha(colour = "black", alpha = 0.05))
  })


  spp_mean <- mean_draws %>% filter(Species == spp)
  mean_line <- plot_kumara(xs = seq(0, 1, length.out = 100),
              a = spp_mean$a, spp_mean$b, spp_mean$c, spp_mean$d, e=spp_mean$e)

  lines(mean_line[[1]], mean_line[[2]],
        lwd = 2, col = colz[state_reg[spp]])

  subEm <- emery %>% filter(Species == spp)
  subEm_0 <- emery_zeros %>% filter(Species == spp)

  points( jitter(subEm$treat, factor = 0.2), subEm$Inflor_biomass,
           pch=19, col=alpha("grey70", 0.75))

  points( jitter(subEm_0$treat, factor = 0.2), subEm_0$Inflor_biomass,
           pch=19, col=alpha("blue", 0.25))


  legend("topleft", spp, bty="n", text.font = 3)

}


axis(side = 2, cex.axis = 0.85, at = round(yrng), las = 1)
axis(side = 1, cex.axis = 0.85)

dev.off()
```

```
## pdf
##   2
```

```
#dev.off()
```

Plotting observed versus predicted

```
meanDraws <- apply(X = posts$mu, 2, mean)

predDF <- data.frame(meanDraws = meanDraws,
                     Inflor_biomass = emery$Inflor_biomass)
```

```
predDF_nZero <- predDF[predDF$Inflor_biomass != 0, ]
predDF_Zero <- predDF[predDF$Inflor_biomass == 0, ]

#plot(emery$Inflor_biomass, meanDraws)
#abline(0,1, lty=2, col="red")

pdf("figures/supp4_fitted_v_resid.pdf")
plot(predDF_nZero$meanDraws, (predDF_nZero$Inflor_biomass - predDF_nZero$meanDraws),
     pch=as.integer(emery$Species), xlab = "fitted values", ylab = "residuals")
points(predDF_Zero$meanDraws, predDF_Zero$Inflor_biomass - predDF_Zero$meanDraws,
     pch="z", cex=1.5)

smth <- smooth.spline(predDF_nZero$meanDraws,
                      predDF_nZero$Inflor_biomass - predDF_nZero$meanDraws)
abline(h=0, lty = 2)
lines(smth$x, smth$y, lty=3, lwd = 2)

legend("topleft", c(as.character(unique(emery$Species))),
       pch = unique(as.integer(emery$Species)),
       ncol = 2, cex =0.4)
dev.off()
```

```
## pdf
##   2
```

```
#plot(predDF_nZero$meanDraws, sqrt((length(predDF_nZero$meanDraws))/sd(predDF_nZero$meanDraws))*(predDF
#points(predDF_Zero$meanDraws, sqrt((length(predDF_Zero$meanDraws))/sd(predDF_Zero$meanDraws))*(predDF_
```

Mean predictions versus smooth spline and mean estimate

```
#meanz <- by(emery$Inflor_biomass,
#            list(emery$treat , emery$Species), mean)

#calculate smooth spline prediction for each treatment and species
smoothed <- group_by(emery, Species) %>%
  do( mod = smooth.spline(.$treat, .$Inflor_biomass))

#name models
names(smoothed$mod) <- smoothed$Species
#get parameters in list
paramz <- as.list(c("a", "b", "c", "d", "e1"))
mean_paramz <- lapply(paramz, function(x){
  apply(posts[[x]], 2, mean)
  }
)

sppint_list <- as.list(1:max(emery$sppint))
xseq <- seq(0,1, length.out=ndraws)
paramz <- as.list(c("a", "b", "c", "d", "e1"))
mean_paramz <- lapply(paramz, function(x){
  apply(posts[[x]], 2, mean)
  }
```

```
)

names(mean_paramz) <- paramz

mean_plot_draws <- lapply(sppint_list, function(sp)
                plot.kumara(xs = xseq,
                            a = mean_paramz$a[sp],
                            b = mean_paramz$b[sp],
                            c = mean_paramz$c[sp],
                            #c = mean_paramz$c[sp]/sppMaxVal[sp], #alt
                            d = mean_paramz$d[sp],
                            e1 = mean_paramz$e1[sp]
                            )
                  )

names(mean_plot_draws) <- unique(emery$Species)

spp_treat_max <- by(emery$treat, emery$Species, max)
kumara_pred <- names(mean_plot_draws) %>% lapply(function(y){
  1:spp_treat_max[y] %>% sapply(function(x){
    mean_plot_draws[[y]][[2]][which.min(abs( mean_plot_draws[[y]][[1]] - x))]
    })
  })

names(kumara_pred) <- names(mean_plot_draws)
smooth_pred <- lapply(smoothed$mod, function(x) x$y)

#sort btoh sets of predicted data
kum_pred_sort <- unlist(kumara_pred)[match(sort(names(unlist(kumara_pred))), names(unlist(kumara_pred))
smooth_pred_sort <- unlist(smooth_pred)[match(sort(names(unlist(smooth_pred))), names(unlist(smooth_pre

namez <- unname(names(kum_pred_sort) %>%  sapply(function(x) unlist(strsplit(x, split = "[[:digit:]]")))

pdf("figures/supp5_model_v_spline.pdf")
plot(kum_pred_sort, smooth_pred_sort,
     pch = as.integer(factor(namez)),
     xlab = "kumaraswamy",
     ylab = "smooth spline",
     main = "predictions from models")
legend("topleft", unique(namez),
       pch = unique(as.integer(factor(namez))), ncol = 3, cex =0.5)

dev.off()
cor(kum_pred_sort, smooth_pred_sort)
```

Plotting parameter densities with priors

```
priorDens <- list(a = NULL, b = NULL, c = NULL,
                  d = NULL, e1 = NULL, nu = NULL,
                  beta_0 = NULL, beta_1 = NULL, lp__ = NULL)
```

```r
nn <- 10000
priorDens$a <- density(rtruncnorm(n = nn, mean = 4, sd = 1, a = 1))
priorDens$b <- density(rtruncnorm(n = nn, mean = 3, sd = 1, a = 1))
priorDens$d <- density(rtruncnorm(nn, mean = min(emery$treat),
                                  sd = 2, b = min(emery$treat)))
priorDens$e1 <- density(rtruncnorm(nn, mean = max(emery$treat),
                                  sd = 2, a = max(emery$treat)))
priorDens$nu <- density(rgamma(nn, shape = 20, scale = 0.2))
priorDens$beta_0 <- density(rnorm(nn, 0, 2))
priorDens$beta_1 <- density(rnorm(nn, 0, 2))

priorDens$c <- density(rtruncnorm( n = nn, mean = 0, sd = 10, a = 0))
hyperDens_c <- density(posts$mean_c)



#!!!!!!
#priorDens$nu <- density(rgamma(nn, shape = 20, scale = 0.2))
#plot(priorDens$nu$x, priorDens$nu$y)
#!!!!!!

pdf("figures/prior_den.pdf")
par(mfrow=c(3,3))
par(mar=c(3,3,2,1))
namez <- c("a", "b", "c", "d", "e", "nu",
           "beta_0", "beta_1", "lp__")
namez_greek <- c(expression(alpha), expression(beta),expression(zeta),
                 expression(delta), expression(epsilon), expression(nu),
                 expression(beta[0]),  expression(beta[1]), "model log posterior")

for(i in 1:length(namez)){
  #i <- 3
  cPost <- posts[[namez[i]]]
  offst <- 0.05
  #if(cPost)
  if(namez[i] == "c"){
    tden <- apply(posts$c, 2, density)
    ylimit <- max(sapply(tden, function(x) max(x$y)*(1 + offst) ))
    xlimit <- range(sapply(tden, function(x) range(x$x)))
    plot(NA,NA, ylim=c(0,ylimit), xlim=xlimit, xlab="", ylab="", main="")
      seq_along(tden) %>%
        lapply(function(x){
         lines(tden[[x]]$x, tden[[x]]$y + (x - 1)*offst)
          #segments(x0 = min(tden[[x]]$x), x1 = max(tden[[x]]$x),
          #         y0 = (x - 1)*offst, y1 = (x - 1)*offst)
        })

      polygon(hyperDens_c, col=alpha("blue", 0.5))
      polygon(priorDens$c, col=alpha("red", 0.5))
      mtext(text = expression(zeta), side = 3)
    } else{
      if(length(dim(cPost)) < 2){
      plot(density(cPost), xlab="", ylab="", main="")
```

```
      polygon(priorDens[[i]], col=alpha("red", 0.5))
      mtext(text = namez_greek[i], side = 3)
    } else{
      tden <- apply(cPost, 2, density)
      ylimit <- max(sapply(tden, function(x) max(x$y)))
      xlimit <- range(sapply(tden, function(x) range(x$x)))
      plot(NA,NA, ylim=c(0,ylimit),
           xlim=xlimit, xlab="", ylab="", main="")
      lapply(tden, lines)
      polygon(priorDens[[i]], col=alpha("red", 0.5))
      mtext(text = namez_greek[i], side = 3)
    }
  }
}

dev.off()
```