Simulated_data_supplement

2022-12-02

```r
library(tidyverse)
```

Fitting the hierarchical, spatial GAMYE model to simulated data of shorebird migration counts

The hierarchical GAMYE model in this paper is based on the model in (Smith and Edwards 2020). The basic model, hierarchical GAM smooth plus year-effects, works well for tracking non-linear patterns in population trajectories.

The shorebird migration monitoring data have some particular biases in space and time, in that many sites and regions have been surveyed for only a portion of the entire 40-year time-series. If they are severe, these spatial and temporal biases and imbalances could confound estimates of population trends.

Here we generated simulated data that exactly match the realized spatial and temporal distribution of counts and surveys in the real monitoring data, but include a known population trend that varies in space. Using these simulated data, we assessed the model's ability to accurately estimate the true simulated trend overall, and in each region. In addition, to specifically test the rigor of one of the key patterns in the real data–that most species of shorebirds show accelerating rates of decline–the simulated data had a constant rate of decline over the entire time period. So in addition to testing if the model could accurately estimate the true rate of decline over the long-term, we also tested if the model could accurately estimate the same long- and short-term trends.

Overall results of simulation

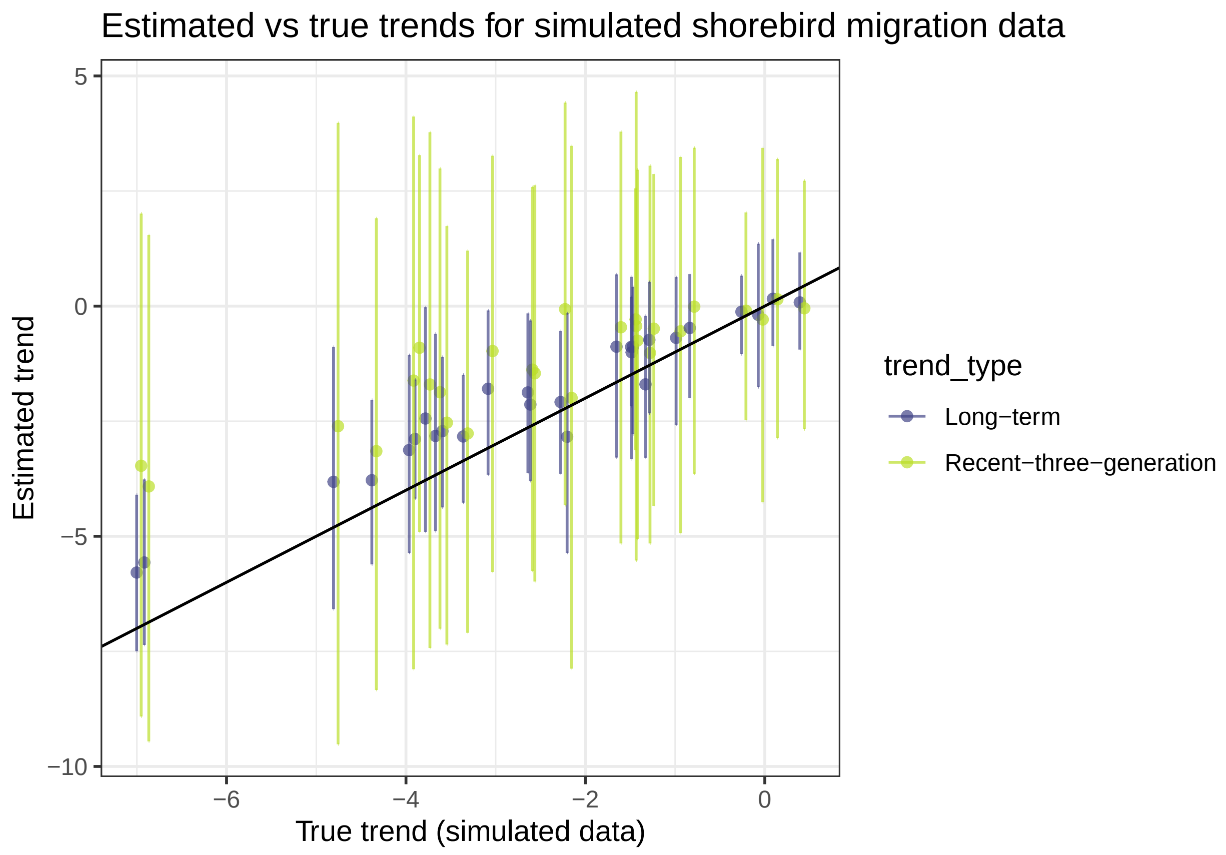
For all species, the model was able to accurately estimate the true underlying trend over the long-term, with some regularizing effect on the more extreme decline estimates. Similarly, the estimated trends for the recent three-generations all have 95% CI that overlap the true trend In comparison to the long-term trend estimates, there is an even stronger regularizing effect for the shorter-term trends. This regularization for extreme long-term trends and for the short-term trends is a desirable feature of this hierarchical model, that guards against making extreme findings of population change without strong evidence. It also makes it clear that there is relatively strong evidence of a recent acceleration in shorebird declines for many species in the real survey data.

```r
sim_T <- read.csv("trends/All_Simulated_gamma-t_survey_wide_trends.csv") %>%
  filter(trend_type %in% c("Long-term","Recent-three-generation"))



trend_comparison_full <- ggplot(data = sim_T,aes(x = true_trend,y = trend,
                                                 colour = trend_type,group = trend_type))+
            xlab("True trend (simulated data)")+
            ylab("Estimated trend")+
            geom_point(position = position_dodge(width = 0.1),alpha = 0.7)+
            geom_errorbar(aes(ymin = lci,ymax = uci),alpha = 0.7,width = 0,
                          position = position_dodge(width = 0.1))+
            geom_abline(intercept = 0,slope = 1)+
  scale_colour_viridis_d(begin = 0.2,end = 0.9)+
  theme_bw()+
  labs(title = "Estimated vs true trends for simulated shorebird migration data")


print(trend_comparison_full)
```



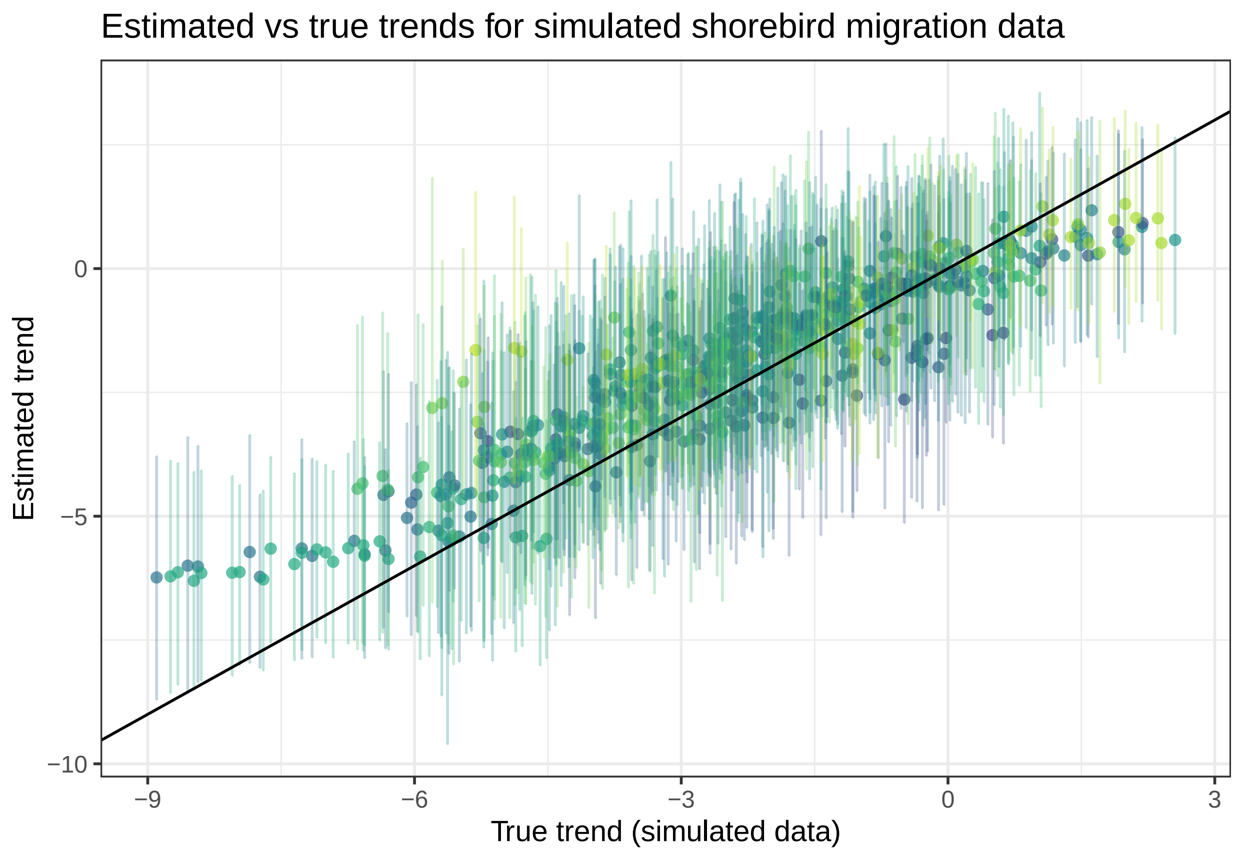
In addition to accurately estimating the simulated survey-wide trends, the model also accurately estimated the strata level trends that varied by latitude. Here again there is evidence of a regularizing effect for the more extreme strata-level trends (e.g., trends < -6%/year), but with the exception of a few of the strata with the most extreme declines, all 95% credible intervals overlap the true simulated strata-level trends.

```
sim_t <- read.csv("trends/All_strata_Simulated_gamma-t_level_trends.csv") %>%
  filter(trend_type %in% c("Long-term"))



trend_comparison_regional <- ggplot(data = sim_t,aes(x = true_trend,y = trend,
                                                     colour = species,group = species))+
            xlab("True trend (simulated data)")+
            ylab("Estimated trend")+
            geom_point(position = position_dodge(width = 0.1),alpha = 0.7)+
          geom_errorbar(aes(ymin = lci,ymax = uci),alpha = 0.3,width = 0,
                        position = position_dodge(width = 0.1))+
          geom_abline(intercept = 0,slope = 1)+
  scale_colour_viridis_d(begin = 0.2,end = 0.9)+
  theme_bw()+
  labs(title = "Estimated vs true trends for simulated shorebird migration data")+
  theme(legend.position = "none")


print(trend_comparison_regional)
```



Simulating the data

Extracting the realized survey data and estimated parameters

To simulate the data, we took the observed surveys for each species (combinations of survey dates and sites) and used the realized estimates of seasonal-effects, site intercepts, and overdispersion to create simulated counts for each survey event from a population with a true log-linear population trend equal to the estimated survey-wide long-term trend for the species. For example, for Red Knot, the simulated data were based on a population that was declining at a constant rate of -6.9%/year, and had the same set of realized monitoring surveys since 1980.

The code below creates the simulated data, fits the model, and then estimates the trends to create the plots above.

```r
library(tidyverse)
library(cmdstanr)
library(shinystan)
library(sf)
library(spdep)
source("functions/utility_functions.R")
source("functions/posterior_summary_functions.R")
source("Functions/palettes.R")
source("functions/GAM_basis_function_mgcv.R")

sps <- readRDS("data/species_vector.rds")

FYYYY = 1980

output_dir <- "output"
#output_dir2 <- "g:/Shorebird_Migration_Trends/output"

# Loading gen times from Bird et al 2020 supplementary material --------
gens = read.csv("data/cobi13486-sup-0004-tables4.csv")
fullgensnames = read.csv("data/cobi13486-sup-0001-tables1.csv")

fullgensnames <- fullgensnames %>% select(Scientific_name,Common_name)
gens <- gens %>% select(Scientific_name,
                        GenLength) %>%
  left_join(.,fullgensnames)

gens[which(gens$Common_name == "American Golden Plover"),"Common_name"] <- "American Golden-Plover"
gens[which(gens$Common_name == "Grey Plover"),"Common_name"] <- "Black-bellied Plover"

#looking for missing species
sps[-which(sps %in% gens$Common_name)]

gens <- gens %>% filter(Common_name %in% sps)


balanced_sim = FALSE # script includes an option to generate a fake dataset that does not have temporal

 for(sp1 in sps){


    spf1 = gsub(sp1,pattern = " ",replacement = "_")
    spf1 = gsub(pattern = "\'",replacement = "",
                x = spf1)
    sp_file_name1 <- paste0(spf1,"-gamma-t")

    load(paste0("data/data",sp1,"_cmdstanr_data.RData"))
    load(paste0(output_dir,"/",sp_file_name1,"_fit_add.RData"))

    cmdstanfit <- rstan::read_stan_csv(csvfl)


# Gather estimates from fitted model --------------------------------------
```

```
    alpha <- posterior_samples(fit = cmdstanfit,
                                           parm = "alpha",
                                           dims = c("s")) %>%
       group_by(s) %>%
       summarise(mean = mean((.value))) %>%
       ungroup() %>%
       select(mean) %>%
       unlist() %>%
       as.numeric()

    ALPHA1 <- posterior_samples(fit = cmdstanfit,
                                 parm = "ALPHA1",
                                 dims = NULL) %>%
       summarise(mean = mean(.value)) %>%
       as.numeric()



    sdnoise <- posterior_samples(fit = cmdstanfit,
                                    parm = "sdnoise",
                                    dims = NULL) %>%
       summarise(mean = mean(.value)) %>%
       as.numeric()

    noise = rnorm(stan_data$ncounts,0,sdnoise)
    # noise2 = rt(stan_data$ncounts,20)*sdnoise
    #
# Seasonal effects --------------------------------------------------------


    if(grepl(x = mod.file2,pattern = "two_season")){
       season_pred_df <- posterior_samples(fit = cmdstanfit,
                                               parm = "season_pred",
                                               dims = c("d","s"))%>%
          group_by(d,s) %>%
          summarise(mean = mean((.value)))

       tmps1 <- season_pred_df %>% filter(s == 1) %>%
          ungroup() %>%
          select(mean) %>%
          unlist() %>%
          as.numeric()
       tmps2 <- season_pred_df %>% filter(s == 2) %>%
          ungroup() %>%
          select(mean) %>%
          unlist() %>%
          as.numeric()

       season_pred <- matrix(c(tmps1,tmps2),ncol = 2)

    }else{
       season_pred_df <- posterior_samples(fit = cmdstanfit,
                                            parm = "season_pred",
```

```r
                                    dims = c("d"))%>%
          group_by(d) %>%
          summarise(mean = mean((.value)))

       season_pred <- season_pred_df %>%
          ungroup() %>%
          select(mean) %>%
          unlist() %>%
          as.numeric()

    }



# fake linear trend to contrast with increasing rate of decline -----------

    year_pred <- matrix(NA,nrow = stan_data$nyears,ncol = stan_data$nstrata)



# use the mean estimated survey wide trend as the base for this sp --------
## then build a trend model that assumes a constant, log-linear slope at that rate
    NSmoothsamples <- posterior_samples(fit = cmdstanfit,
                                    parm = "NSmooth",
                                    dims = c("y"))
    NSmoothsamples$year <- NSmoothsamples$y + (1980-1)


    t_NSmooth_80 <- ItoT(inds = NSmoothsamples,
                         start = 1980,
                         end = 2019,
                         regions = NULL,
                         qs = 95,
                         sp = sp,
                         type = "Long-term")


    B1 = 0.5*(t_NSmooth_80$trend/100)  #linear trend at center of range = 2*B1
    bx = -0.005 #implies a 1-2%/year range in trends by longitude
    by = -0.02 #implies a 2-4%/year range in trends by latitude

    strats_xy <- strats_dts %>%
       mutate(x = as.numeric(str_split(hex_name,pattern = "_",simplify = TRUE)[,1]),
              y = as.numeric(str_split(hex_name,pattern = "_",simplify = TRUE)[,2]),
              x_scale = (x-mean(x))/(0.5*diff(range(x))),#scaled x coordinate to create a longitudinal
              y_scale = (y-mean(y))/(0.5*diff(range(y))),#scaled y coordinate to create a latitudinal g
              b1 = (x_scale*bx+B1) + (y_scale*by+B1)) #initial log-linear slopes for each stratum with

    midyear = 20
    b1 = strats_xy$b1 #initial slopes for each stratum

    #generating a smooth log-linear trajectory for each stratum
    for(s in 1:stan_data$nstrata){
       year_pred[1:midyear,s] <- b1[s]*((1:midyear)-midyear)
       year_pred[(midyear+1):stan_data$nyears,s] <- b1[s]*((midyear+1):stan_data$nyears-midyear)
    }



# fake - evenly distributed and oversimplified year-effects ----------------------------------
      # the even distribution and simple, systematic year-effects
      # ensure that there's no accidental pattern in the year-effects that might
      # induce bias in estimated smooth component that tracks the species trend

    sdyear <- posterior_samples(fit = cmdstanfit,
                                parm = "sdyear",
                                dims = NULL) %>%
       summarise(mean = mean(.value)) %>%
       as.numeric()

    year_effect <- rep(c(-1,1),length = stan_data$nyears)*(sdyear/2)
    year_effect[c(1,stan_data$nyears)] <- 0





# Generate fake counts using above estimates ------------------------------

    stan_data_sim = stan_data

    if(grepl(x = mod.file2,pattern = "two_season")){
    for(i in 1:stan_data_sim$ncounts){
       stan_data_sim$count[i] =
         rpois(n = 1,exp(ALPHA1 +
                           year_pred[stan_data_sim$year_raw[i],stan_data_sim$strat[i]] +
                           alpha[stan_data_sim$site[i]] +
                           year_effect[stan_data_sim$year_raw[i]] +
                           season_pred[stan_data_sim$date[i],stan_data_sim$seas_strat[i]] +
                           noise[i]));

    }
    }else{
       for(i in 1:stan_data_sim$ncounts){
          stan_data_sim$count[i] =
            rpois(n = 1,exp(ALPHA1 +
                              year_pred[stan_data_sim$year_raw[i],stan_data_sim$strat[i]] +
                              alpha[stan_data_sim$site[i]] +
                              year_effect[stan_data_sim$year_raw[i]] +
                              season_pred[stan_data_sim$date[i]] + noise[i]));
       }
    }



    # catch a few extreme counts -----------------------------------
```

```
    tt = (which(stan_data_sim$count > max(stan_data$count)*2))
    if(length(tt)/stan_data$ncounts > 0.002){
      warning(paste(sp,"simulated data has many extreme values"))
    }
    stan_data_sim$count[tt] <- max(stan_data$count)*2

# compare fake to observed data -------------------------------------------




    comp_dat1 = data.frame(year_raw = stan_data$year_raw,
                          site = stan_data$site,
                          strat = as.factor(stan_data$strat),
                          date = stan_data$date,
                          count = stan_data$count,
                          sim = "observed")

    comp_dat2 = data.frame(year_raw = stan_data$year_raw,
                           site = stan_data$site,
                           strat = as.factor(stan_data$strat),
                           date = stan_data$date,
                           count = stan_data_sim$count,
                           sim = "simulated")
    comp_dat <- bind_rows(comp_dat1,comp_dat2)

    #remove the fitted object from the real-data analysis
rm(list = "cmdstanfit")

# fit simulated data ------------------------------------------------------


sp = paste("Simulated",sp1,sep = "_")


   spf = gsub(sp,pattern = " ",replacement = "_")
   spf = gsub(pattern = "\'",replacement = "",
           x = spf)
   sp_file_name <- paste0(spf,"-gamma-t")

   pdim = ceiling(sqrt(stan_data$nstrata)) # just for plotting

   pdf(paste0("figures/",sp_file_name,"_sim_vs_obs.pdf"),
       width = 11,
       height = 8)
   # comp_box = ggplot(data = comp_dat,aes(y = count,colour = sim))+
   #    geom_boxplot()+
   #    facet_wrap(~strat,nrow = pdim, ncol = pdim,scales = "free")
   #
   # print(comp_box)

   comp_point = ggplot(data = comp_dat,aes(y = count+1,x = sim,colour = sim))+
      geom_point(position = position_jitter(width = 0.4))+
      scale_y_log10()+
      facet_wrap(~strat,nrow = pdim, ncol = pdim,scales = "free")

   print(comp_point)

   dev.off()



   #if(!file.exists(paste0(output_dir,"/",sp_file_name,"-",1,".csv"))){






    ## compile model
    modl = cmdstan_model(stan_file=mod.file2)

print(sp)

if(grepl(x = mod.file2,pattern = "two_season")){
init_def <- function(){ list(noise_raw = rnorm(stan_data$ncounts,0,0.1),
                             alpha_raw = rnorm(stan_data$nsites,0,0.1),
                             ALPHA1 = 0,
                             year_effect_raw = rnorm(stan_data$nyears,0,0.1),
                             B_season_raw1 = rnorm(stan_data$ndays,0,0.1),
                             B_season_raw2 = rnorm(stan_data$ndays,0,0.1),
                             sdnoise = 0.2,
                             sdalpha = 0.1,
                             sdyear_gam = 1,
                             sdyear_gam_strat = runif(stan_data$nknots_year,0.1,0.2),
                             sdseason = c(0.1,0.1),
                             sdyear = 0.1,
                             B_raw = rnorm(stan_data$nknots_year,0,0.1),
                             b_raw = matrix(rnorm(stan_data$nknots_year*stan_data$nstrata,0,0.01),
                                            nrow = stan_data$nstrata,ncol = stan_data$nknots_year))}

}else{
   init_def <- function(){ list(noise_raw = rnorm(stan_data$ncounts,0,0.1),
                                alpha_raw = rnorm(stan_data$nsites,0,0.1),
                                ALPHA1 = 0,
                                year_effect_raw = rnorm(stan_data$nyears,0,0.1),
                                B_season_raw = rnorm(stan_data$ndays,0,0.1),
                                #B_season_raw2 = rnorm(stan_data$ndays,0,0.1),
                                sdnoise = 0.2,
                                sdalpha = 0.1,
                                sdyear_gam = 1,
                                sdyear_gam_strat = runif(stan_data$nknots_year,0.1,0.2),
                                sdseason = c(0.1),
                                sdyear = 0.1,
                                B_raw = rnorm(stan_data$nknots_year,0,0.1),
                                b_raw = matrix(rnorm(stan_data$nknots_year*stan_data$nstrata,0,0.01),
                                                nrow = stan_data$nstrata,ncol = stan_data$nknots_year))}
}

cmdstanfit<- modl$sample(data=stan_data_sim,
               refresh=100,
               chains=4, iter_sampling =800,
               iter_warmup=1000,
               parallel_chains = 4,
               max_treedepth = 15,
               adapt_delta = 0.8,
               init = init_def)


cmdstanfit$save_output_files(dir = output_dir,
                             basename = sp_file_name,
                             timestamp = FALSE,
                             random = FALSE)

csvfl = paste0(output_dir,"/",sp_file_name,"-",1:4,".csv")
#cmdstanfit$save_object(file = paste0(output_dir,"/",sp_file_name,".RDS"))

shiny_explore <- FALSE
if(shiny_explore){
   #load(paste0(output_dir,"/",sp_file_name,"_fit_add.RData"))
  sl_rstan <- rstan::read_stan_csv(csvfl)
  launch_shinystan(as.shinystan(sl_rstan))
}




save(list = c("cmdstanfit",
              "stan_data_sim",
              "dts",
              "real_grid",
              "strats_dts",
              "strat_regions",
              "parms",
              "sp",
              "spf",
              "sp_file_name",
              "output_dir",
              "prior",
              "mod.file2",
              "noise_dist2",
              "csvfl",
              "strats_xy"),
     file = paste0(output_dir,"/",sp_file_name,"_fit_add.RData"))
```

```r
if(balanced_sim){
# Fit simulated data with balanced sampling -------------------------------




stan_data_bal <- stan_data

#table of the sites to sample
sites_strats <- unique(data.frame(strat = stan_data$strat,
                                  site = stan_data$site,
                                  seas_strat = stan_data$seas_strat))

# balanced sequence of sampling within season - assuming each site sample 12 times each year
sample_days = floor(seq(6,stan_data$ndays-6,length = 12))

bal_days <- NULL
for(j in 1:nrow(sites_strats)){ #adding some random variation to which days each site is sampled
   tmp = sites_strats[j,]
   tmp2 = sample_days + round(runif(length(sample_days),-5,5))
   tmp = expand_grid(tmp,date = as.integer(tmp2))

   bal_days <- bind_rows(bal_days,tmp)
}

bal_dat <- expand_grid(bal_days,year_raw = c(1:stan_data$nyears))


stan_data_bal$ncounts <- nrow(bal_dat)
stan_data_bal$year_raw <- bal_dat$year_raw
stan_data_bal$site <- bal_dat$site
stan_data_bal$strat <- bal_dat$strat
stan_data_bal$date <- bal_dat$date
stan_data_bal$seas_strat <- bal_dat$seas_strat


noise = rnorm(stan_data_bal$ncounts,0,sdnoise)

stan_data_bal$count <- vector(mode = "integer",length = stan_data_bal$ncounts)


# Generate fake counts using above estimates ------------------------------



if(grepl(x = mod.file2,pattern = "two_season")){
   for(i in 1:stan_data_sim$ncounts){
      stan_data_bal$count[i] =
        rpois(n = 1,exp(ALPHA1 +
                          year_pred[stan_data_bal$year_raw[i],stan_data_bal$strat[i]] +
                          alpha[stan_data_bal$site[i]] +
                          year_effect[stan_data_bal$year_raw[i]] +
                          season_pred[stan_data_bal$date[i],stan_data_bal$seas_strat[i]] +
                          noise[i]));

   }
}else{
   stan_data_bal$count[i] =
     rpois(n = 1,exp(ALPHA1 +
                        year_pred[stan_data_bal$year_raw[i],stan_data_bal$strat[i]] +
                        alpha[stan_data_bal$site[i]] +
                        year_effect[stan_data_bal$year_raw[i]] +
                        season_pred[stan_data_bal$date[i]] + noise[i]));

}



# catch a few extremely unlikely counts -----------------------------------

tt = (which(stan_data_bal$count > max(stan_data$count)*2))
if(length(tt)/stan_data$ncounts > 0.002){
   warning(paste(sp,"simulated data have many extreme values"))
}
stan_data_bal$count[tt] <- max(stan_data$count)*2



# fit Balanced simulated data -----------------------------------------------------


#sp = paste("Simulated",sp1,sep = "_")
sp = paste0("Balanced_",sp)


spf = gsub(sp,pattern = " ",replacement = "_")
spf = gsub(pattern = "\'",replacement = "",
           x = spf)
sp_file_name <- paste0(spf,"-gamma-t")


#if(!file.exists(paste0(output_dir,"/",sp_file_name,"-",1,".csv"))){






## compile model
modl = cmdstan_model(stan_file=mod.file2)

print(sp)

if(grepl(x = mod.file2,pattern = "two_season")){

init_def <- function(){ list(noise_raw = rnorm(stan_data_bal$ncounts,0,0.1),
                             alpha_raw = rnorm(stan_data_bal$nsites,0,0.1),
                             ALPHA1 = 0,
                             year_effect_raw = rnorm(stan_data_bal$nyears,0,0.1),
                             B_season_raw1 = rnorm(stan_data_bal$ndays,0,0.1),
                             B_season_raw2 = rnorm(stan_data_bal$ndays,0,0.1),
                             sdnoise = 0.2,
                             sdalpha = 0.1,
                             sdyear_gam = 1,
                             sdyear_gam_strat = runif(stan_data_bal$nknots_year,0.1,0.2),
                             sdseason = c(0.1,0.1),
                             sdyear = 0.1,
                             B_raw = rnorm(stan_data_bal$nknots_year,0,0.1),
                             b_raw = matrix(rnorm(stan_data_bal$nknots_year*stan_data_bal$nstrata,0,0.0
                                            nrow = stan_data_bal$nstrata,ncol = stan_data_bal$nknots_yea


}else{
   init_def <- function(){ list(noise_raw = rnorm(stan_data_bal$ncounts,0,0.1),
                                alpha_raw = rnorm(stan_data_bal$nsites,0,0.1),
                                ALPHA1 = 0,
                                year_effect_raw = rnorm(stan_data_bal$nyears,0,0.1),
                                B_season_raw = rnorm(stan_data_bal$ndays,0,0.1),
                                #B_season_raw2 = rnorm(stan_data_bal$ndays,0,0.1),
                                sdnoise = 0.2,
                                sdalpha = 0.1,
                                sdyear_gam = 1,
                                sdyear_gam_strat = runif(stan_data_bal$nknots_year,0.1,0.2),
                                sdseason = c(0.1),
                                sdyear = 0.1,
                                B_raw = rnorm(stan_data_bal$nknots_year,0,0.1),
                                b_raw = matrix(rnorm(stan_data_bal$nknots_year*stan_data_bal$nstrata,0,
                                               nrow = stan_data_bal$nstrata,ncol = stan_data_bal$nknots_

}

cmdstanfit<- modl$sample(data=stan_data_bal,
                         refresh=100,
                         chains=4, iter_sampling =800,
                         iter_warmup=1000,
                         parallel_chains = 4,
                         max_treedepth = 15,
                         adapt_delta = 0.8,
                         init = init_def)


cmdstanfit$save_output_files(dir = output_dir,
                             basename = sp_file_name,
                             timestamp = FALSE,
                             random = FALSE)

csvfl = paste0(output_dir,"/",sp_file_name,"-",1:4,".csv")
#cmdstanfit$save_object(file = paste0(output_dir,"/",sp_file_name,".RDS"))

shiny_explore <- FALSE
if(shiny_explore){
   #load(paste0(output_dir,"/",sp_file_name,"_fit_add.RData"))
   sl_rstan <- rstan::read_stan_csv(csvfl)
   launch_shinystan(as.shinystan(sl_rstan))
}




save(list = c("cmdstanfit",
              "stan_data_bal",
              "dts",
              "real_grid",
              "strats_dts",
              "strat_regions",
              "parms",
              "sp",
              "spf",
              "sp_file_name",
              "output_dir",
              "prior",
              "mod.file2",
              "noise_dist2",
              "csvfl",
              "strats_xy"),
     file = paste0(output_dir,"/",sp_file_name,"_fit_add.RData"))






}#end if balanced


 }### end of species loop









# COMPARE FITTED WITH KNOWN ------------------------------------------

#output_dir <- "g:/Shorebird_Migration_Trends/output"

output_dir <- "output"

source("functions/utility_functions2.R") #loads alternate indices function
```

```
trendsout <- NULL
TRENDSout <- NULL

saved_trends <- read.csv("trends/All_gamma_t_survey_wide_trends.csv")

# Comarison Species Loop
for(sp1 in sps){





saved_trend <- saved_trends[which(saved_trends$species == sp1 &
                                     saved_trends$start_year == 1980 &
                                     saved_trends$end_year == 2019), "trend"]


   sp = paste("Simulated",sp1,sep = "_")



   spf = gsub(sp,pattern = " ",replacement = "_")
   spf = gsub(pattern = "\'",replacement = "",
              x = spf)
   sp_file_name <- paste0(spf,"-gamma-t")




      if(file.exists(paste0(output_dir,"/",sp_file_name,"_fit_add.RData"))){
         load(paste0(output_dir,"/",sp_file_name,"_fit_add.RData"))
        output_dir <- "output"

        dts$count <- stan_data_sim$count


         B1 = 0.5*(saved_trend/100)  #linear trend at center of range = 2*B1
         bx = -0.005 #implies a 1-2%/year range in trends by longitude
         by = -0.02 #implies a 2-4%/year range in trends by latitude

         strats_xy <- strats_dts %>%
            mutate(x = as.numeric(str_split(hex_name,
                                            pattern = "_",simplify = TRUE)[,1]),
                   y = as.numeric(str_split(hex_name,
                                            pattern = "_",simplify = TRUE)[,2]),
                   x_scale = (x-mean(x))/(0.5*diff(range(x))),
                   #scaled x coordinate to create a longitudinal gradient in trends
                   y_scale = (y-mean(y))/(0.5*diff(range(y))),
                   #scaled y coordinate to create a latitudinal gradient in trends
                   b1 = (x_scale*bx+B1) + (y_scale*by+B1))
         #initial log-linear slopes for each stratum with spatial gradients

         midyear = 20
```

```
b1 = strats_xy$b1 #initial slopes for each stratum
#b2 = strats_xy$b2 #second slopes for each stratum


if(length(csvfl) > 4){csvfl <- csvfl[1:4]}
csvfl2 <- unlist(str_split(csvfl,pattern = "/"))
csvfl <- paste0(output_dir,"/",csvfl2[grepl(csvfl2,pattern = ".csv",
                                            fixed = TRUE)])
cmdstanfit <- cmdstanr::as_cmdstan_fit(csvfl)
three_gen <- max(10,ceiling(gens[which(gens$Common_name == sp),"GenLength"]*3)) #20
#Three generation assessment time in COSEWIC report
y3g <- 2019-three_gen



# Calculate Annual indices using samples ----------------------------------

syear = min(dts$YearCollected)




NSmoothsamples <- posterior_samples(fit = cmdstanfit,
                                    parm = "NSmooth",
                                    dims = c("y"))
NSmoothsamples$year <- NSmoothsamples$y + (syear-1)



 sites_strat = (stan_data_sim$sites)
nstrata = stan_data_sim$nstrata
nsites_strat = stan_data_sim$nsites_strat



# calculate trends continent --------------------------------------------------------

t_NSmooth_80 <- ItoT(inds = NSmoothsamples,
                     start = 1980,
                     end = 2019,
                     regions = NULL,
                     qs = 95,
                     sp = sp,
                     type = "Long-term")



t_NSmooth_3g <- ItoT(inds = NSmoothsamples,
                     start = y3g,
                     end = 2019,
                     regions = NULL,
                     qs = 95,
                     sp = sp,
                              type = "Recent-three-generation")


syL3g = y3g-(2019-y3g)
syL3g = max(1980,syL3g)
t_NSmooth_L3g <- ItoT(inds = NSmoothsamples,
                      start = syL3g,
                      end = y3g,
                      regions = NULL,
                      qs = 95,
                      sp = sp,
                      type = "Previous-three-generation")



nstrata = stan_data_sim$nstrata

nsmoothsamples <- posterior_samples(fit = cmdstanfit,
                                    parm = "nsmooth",
                                    dims = c("stratn","y"))


nsmoothsamples$year <- nsmoothsamples$y + (syear-1)
nsmoothsamples <- left_join(nsmoothsamples,strats_dts,by = c("stratn"))



t_nsmooth_strat_80 <- ItoT(inds = nsmoothsamples,
                           start = 1980,
                           end = 2019,
                           regions = "hex_name",
                           qs = 95,
                           sp = sp,
                           type = "Long-term",
                           centered_trends = TRUE)


t_nsmooth_strat_3g <- ItoT(inds = nsmoothsamples,
                           start = y3g,
                           end = 2019,
                           regions = "hex_name",
                           qs = 95,
                           sp = sp,
                           type = "Recent-three-generation",
                           centered_trends = TRUE)

t_nsmooth_strat_L3g <- ItoT(inds = nsmoothsamples,
                            start = syL3g,
                            end = y3g,
                            regions = "hex_name",
                            qs = 95,
                            sp = sp,
                            type = "Previous-three-generation",
                                     centered_trends = TRUE)









  strats_xy$true_trend <- 100*(exp(strats_xy$b1)-1)
  strats_xy <- strats_xy %>%
    select(hex_name,true_trend)



tcomp_80 <-  t_nsmooth_strat_80 %>%
     left_join(.,strats_xy,by = c("region" = "hex_name")) %>%
     mutate(prec = 1/((uci-lci)/(1.96*2))^2)

  t_Ncomp_80 <- t_NSmooth_80 %>%
    mutate(region = "Survey Wide",
           true_trend = saved_trend)
  t_Ncomp_3g <- t_NSmooth_3g %>%
    mutate(region = "Survey Wide",
           true_trend = saved_trend)
  t_Ncomp_L3g <- t_NSmooth_L3g %>%
    mutate(region = "Survey Wide",
           true_trend = saved_trend)

  tcomp_3g <- left_join(t_nsmooth_strat_3g,strats_xy,by = c("region" = "hex_name")) %>%
     mutate(prec = 1/((uci-lci)/(1.96*2))^2)



  tcomp_L3g <- left_join(t_nsmooth_strat_L3g,strats_xy,by = c("region" = "hex_name")) %>%
     mutate(prec = 1/((uci-lci)/(1.96*2))^2)


    TRENDSout <- bind_rows(TRENDSout,t_Ncomp_80)
    TRENDSout <- bind_rows(TRENDSout,t_Ncomp_3g)
    TRENDSout <- bind_rows(TRENDSout,t_Ncomp_L3g)


    trendsout <- bind_rows(trendsout,
                           tcomp_80)

    trendsout <- bind_rows(trendsout,
                           tcomp_3g)
    trendsout <- bind_rows(trendsout,
                           tcomp_L3g)
```

```
        print(sp)

      }# end if species output data exists
}   #end species loop




write.csv(trendsout,paste0("trends/All_region_Simulated_strata_","gamma-t","_composite_trends.csv"),row
write.csv(TRENDSout,paste0("trends/All_Simulated_","gamma-t","_survey_wide_trends.csv"),row.names = FALS
```

Smith, Adam C., and Brandon P. M. Edwards. 2020. "North American Breeding Bird Survey Status and Trend Estimates to Inform a Wide-Range of Conservation Needs, Using a Flexible Bayesian Hierarchical Generalized Additive Model." *bioRxiv*, October, 2020.03.26.010215. https://doi.org/10.1101/2020.03.26.010215.