A sample workflow

Victor Jonathan Lizzie

April 2025

Table of contents

To show our workflow in action, we examine a subset of the Living Planet Index (LPI, download at: https://www.livingplanetindex.org/data_portal. This download should include a csv file, here: LPD_2024_public.csv, that we use below). We discuss the LPI in the main text; for more information, check out their website.

1 Step 1: Research question

Here we need to focus on exactly what question we are trying to answer, which will guide us in the next steps. This step is sometimes skipped over or moved through so quickly that the

exact question is not clear, but is critical for developing a useful model. Many early papers on the LPI seem focused on something similar to: Are vertebrate species overall declining on average? And, if so, by how much? We, however, might want to have a question that more clearly allows any directionality, so we could start with: What is the global trend for biodiversity in vetebrate species? This is rather broad and might mean we should develop a model that will yield one global estimate. In thinking about this, we would likely realize we don't want just that, we actually want to understand the variation across different species and perhaps populations and also have a global estimate. So we might refine the question to:

> What is the global trend over time in vertebrate species population numbers, both overall and how does it vary across species and populations?

```
wd <- "/home/victor/projects/forecastflows/analyses"

set.seed(20012029)

suppressPackageStartupMessages(library(rstan))
library(ggplot2)

lpi_data <- read.csv(file.path(wd, "input", "LPD_2024_public.csv"), header=T)
```

2 Step 2: Model building

Next we need to think about how we would build a model to address this question, with the goal to build the model and evaluate it using simulated data. There are a lot of demographic models in population ecology, so we might revisit those and consider a discrete time model based on population size and stepping through each year of data. We might also start with a simple linear model, where populations can go up, down or do nothing.

Using a simple linear gets us started, but now we need to consider how we will model species and populations. Populations usually are nested in species in ecological models, so we might consider that to start. In this first hierarchical model, we thus assume that populations are exchangeable—an assumption we might refine later. We also need to make sure our model yields one global estimate; for this we decide to model species trends over time as part of larger distribution. Our resulting model would then be:

$$\log(y_i) \sim \text{normal}(\mu_i,\ \sigma)$$

$$\mu_i = \alpha_{\text{pop[sp[}i\text{]]}} + \beta_{\text{pop[sp[}i\text{]]}} * \text{year}_i$$

$$\alpha_{\text{pop[sp]}} \sim \text{normal}(\mu_{\alpha,\text{sp}},\ \sigma_{\alpha,\text{sp}}) \quad \mu_{\alpha,\text{sp}} \sim \text{normal}(\mu_{\alpha_1},\ \sigma_{\alpha_1}) \quad \sigma_{\alpha,\text{sp}} \sim \text{normal}(\mu_{\alpha_2},\ \sigma_{\alpha_2})$$

$$\beta_{\text{pop[sp]}} \sim \text{normal}(\mu_{\beta,\text{sp}},\ \sigma_{\beta,\text{sp}}) \quad \mu_{\beta,\text{sp}} \sim \text{normal}(\mu_{\beta_1},\ \sigma_{\beta_1}) \quad \sigma_{\beta,\text{sp}} \sim \text{normal}(\mu_{\beta_2},\ \sigma_{\beta_2})$$

(where all the σ parameters are greater than zero)

3 Step 3: Model evaluation

Now we need to simulate data to test our model. As we start to think about how to do this, we may become overwhelmed with the scale of the question we started with. We likely would not feel immediately sure how to simulate for all different vertebrate species. For example, what are reasonable starting population sizes (intercepts) for different groups? As we start to answer this we likely would realize different clades should have different starting population sizes. We could thus try to build in increasing complexity immediately, but then we will likely struggle to see how our model is performing at the scales we care about – for each population in each species. It is almost always better to scale back the question (and potentially dataset) when confronted with these sorts of problems.

3.1 Stepping backward...

Here we would likely feedback (step back) one to two steps. We could step back to Step 2 and consider adding a phylogeny or other structure from which we could then simulate data from, or we could step back to Step 1 and narrow the question (before potentially building it back up.) As mentioned above, we believe the latter approach is usually better as working on a small scale first can point out more fundamental problems versus starting big and then spending much longer to identify fundamental problems (or potentially missing them).

So here we refine our question to:

> What is trend in mammal species population numbers in southern Africa, both overall and how does it vary across species and populations?

We chose this because mammals are a well studied group compared to other clades/groups in the dataset, so we will have a better idea of how to simulate data, and whether our model makes sense.

```
lpi_subset <-subset(lpi_data, Region == "Africa" & Class == "Mammalia" &
                      Included.in.LPR2024 == 1 )

# subset 5+ counts
obs.cnt<-NULL
for (n in 1:dim(lpi_subset)[1]){
  obs.cnt[n]<-sum(lpi_subset[n, 31:101]!="NULL")
}
lpi_subset<-lpi_subset[obs.cnt>4,]

# subset species 10+ populations
sp <- unique(lpi_subset$Binomial)
sp.cnt<-NULL
for(x in 1:length(sp)){
  Af.tmp<-subset(lpi_subset, Binomial == sp[x])
  sp.cnt[x]<-dim(Af.tmp)[1]
}
keep.sp<-sp[sp.cnt>9]
lpi_subset<-lpi_subset[lpi_subset$Binomial %in% keep.sp,]
```

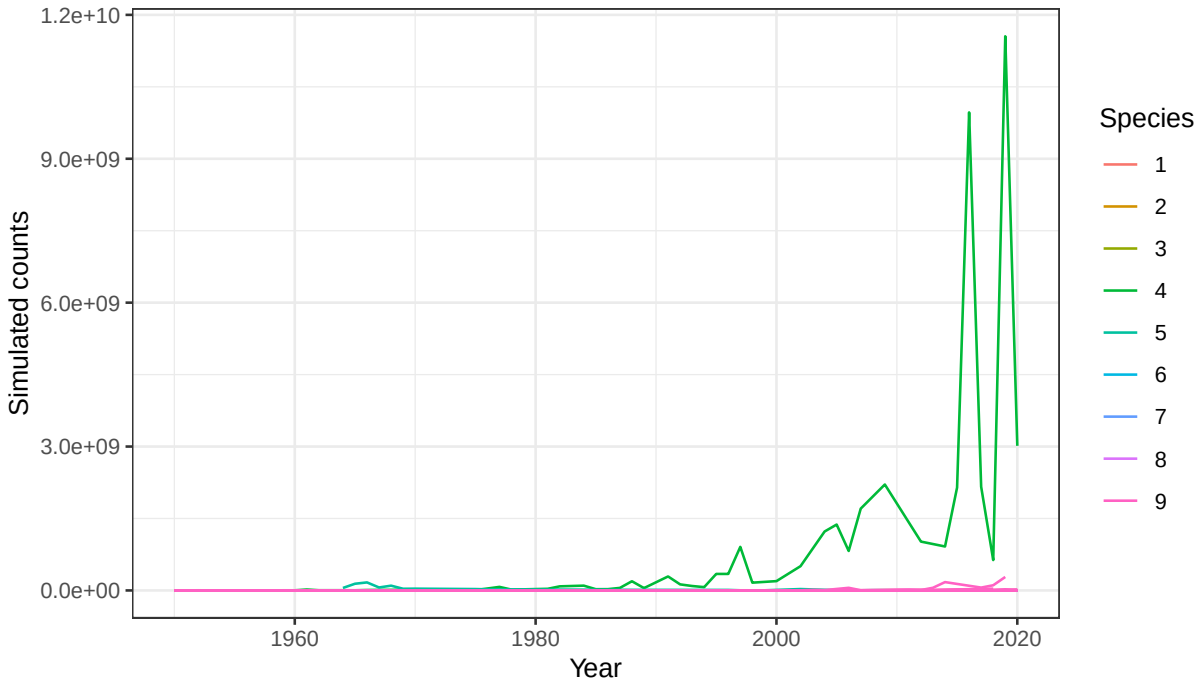
3.2 Simulating data

Now we start to simulate! We follow the mathematical model we built above and simulate data from it.

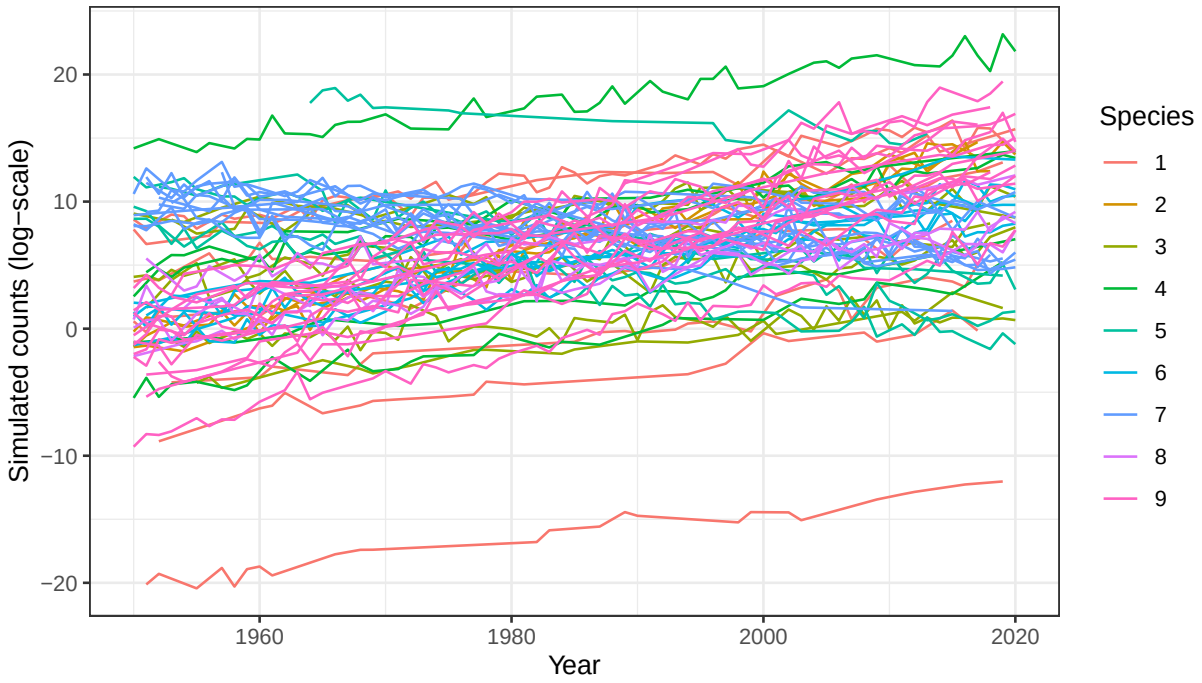
```
nspecies <- 9 # no. of species
npops_persp <- round(runif(nspecies, 2, 15)) # no. of different populations per species
npops <- sum(npops_persp) # total no. of different populations
nobs_perpop <- round(runif(npops, 20, 60)) # no. of different observations per population

spid_perpop <- rep(1:nspecies, times = npops_persp)
spid_perobs <- rep(spid_perpop, times = nobs_perpop)
popid_perobs <- rep(1:npops, times = nobs_perpop)

years <- c()
for(np in nobs_perpop){
  years_p <- sample(1950:2020, size = np, replace = FALSE)
  years <- c(years, years_p)
}

# nested intercepts
```

```r
mu_alpha1 <- log(200)
sigma_alpha1 <-  log(50)/2.57
mu_alpha_sp <- rnorm(n = nspecies, mean = mu_alpha1, sd = sigma_alpha1)
mu_alpha2 <- log(20)
sigma_alpha2 <- log(50)/2.57
sig_species_alpha <- abs(rnorm(n = nspecies, mean = mu_alpha2,
                               sd = sigma_alpha2))
alpha_pop_sp <- rnorm(n = npops, mean = mu_alpha_sp[spid_perpop],
                      sd = sig_species_alpha[spid_perpop])

# nested slopes
mu_beta1 <- 0.05
sigma_beta1 <- 0.25/2.57
mu_beta_sp <- rnorm(n = nspecies, mean = mu_beta1, sd = sigma_beta1)
mu_beta2 <- 0
sigma_beta2 <- 0.05/2.57
sig_species_beta <- abs(rnorm(n = nspecies, mean = mu_beta2,
                              sd = sigma_beta2))
beta_pop_sp <- rnorm(n = npops, mean = mu_beta_sp[spid_perpop],
                     sd = sig_species_beta[spid_perpop])

# save simulated intercepts and slopes
pops_fit <- data.frame()
pid <- 1
for(sp in 1:nspecies){
  for(p in 1:npops_persp[sp]){
    pops_fit <- rbind(pops_fit, data.frame(species = sp, pop = pid))
    pid <- pid + 1
  }
}
pops_fit$int <- alpha_pop_sp
pops_fit$slp <- beta_pop_sp

sigma_obs <- 0.75 # common observational error

y <- c()
for(p in 1:npops){
  nobs_p <- nobs_perpop[p]
  alpha_p <- alpha_pop_sp[p]
  beta_p <- beta_pop_sp[p]

  years_p <- years[popid_perobs == p]
```

```
  yhat_p <- alpha_p + beta_p * (years_p - 1980)
  # yhat_p <- alpha_p
  y_p <- exp(rnorm(n = nobs_p, mean = yhat_p, sd = sigma_obs))
  y <- c(y, y_p)
}


datasim <- data.frame(
  y,
  year = years - 1980,
  species = as.character(spid_perobs),
  population = paste0(spid_perobs,popid_perobs)
)

datasim$logy <- log(datasim$y)

ggplot(data = datasim) +
  geom_line(aes(x = year+1980, y = y,
                group = population,
                color = species)) +
  theme_bw() +
  labs(x = 'Year', y = 'Simulated counts', colour = 'Species')
```



```
ggplot(data = datasim) +
  geom_line(aes(x = year+1980, y = logy,
                group = population,
                color = species)) +
  theme_bw() +
  labs(x = 'Year', y = 'Simulated counts (log-scale)', colour = 'Species')
```



3.3 Fit and check model on simulated data!

Now we code the model from our mathematical notation (above) and fit it to our simulated data (also built from our mathematical model).

```
data {
  int<lower=1> N;
  int<lower=1> Nsp;
  int<lower=1> Npop;
  vector[N] y;
  vector[N] year;
  array[Npop] int<lower=1,upper=Nsp> spid_perpop;
  array[N] int<lower=1,upper=Npop> popid;
}
```

```
transformed data{
  vector[N] logy = log(y);
}

parameters {
  real mu_alpha1; // overall species intercept
  real<lower=0> sigma_alpha1;

  real mu_alpha2;
  real<lower=0> sigma_alpha2;

  array[Nsp] real mu_alpha_sp; // species-level intercepts
  array[Nsp] real<lower=0> sig_species_alpha;

  array[Npop] real alpha_tilde_pop_sp; // non-centered population intercepts

  real mu_beta1; // overall species trend
  real<lower=0> sigma_beta1;

  real mu_beta2;
  real<lower=0> sigma_beta2;

  array[Nsp] real mu_beta_sp; // species-level slopes
  array[Nsp] real<lower=0> sig_species_beta;

  array[Npop] real beta_tilde_pop_sp; // non-centered population slopes

  real<lower=0> sigma;

}

transformed parameters{

  array[Npop] real alpha_pop_sp;
  array[Npop] real beta_pop_sp;

  for (p in 1:Npop) {

      alpha_pop_sp[p] = mu_alpha_sp[spid_perpop[p]]
      + sig_species_alpha[spid_perpop[p]] *  alpha_tilde_pop_sp[p];
      beta_pop_sp[p] = mu_beta_sp[spid_perpop[p]]
      + sig_species_beta[spid_perpop[p]] *  beta_tilde_pop_sp[p];
```

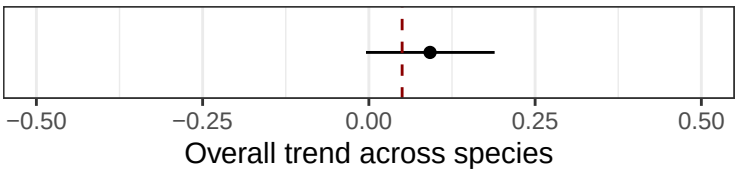
```
  }

}

model {

  vector[N] mu;

  mu_alpha_sp ~ normal(mu_alpha1, sigma_alpha1);
  sig_species_alpha ~ normal(mu_alpha2, sigma_alpha2);

  mu_beta_sp ~ normal(mu_beta1, sigma_beta1);
  sig_species_beta ~ normal(mu_beta2, sigma_beta2);

  for (p in 1:Npop) {

    alpha_tilde_pop_sp[p] ~ normal(0,1);
    beta_tilde_pop_sp[p] ~ normal(0,1);

  }

  for(i in 1:N){

    mu[i] = alpha_pop_sp[popid[i]] + beta_pop_sp[popid[i]] * (year[i] - 1980);
    logy[i] ~ normal(mu[i], sigma);

  }

  mu_alpha1 ~ normal(0, log(500) / 2.57); // -log(500) < mu_alpha1 < log(500)
  sigma_alpha1 ~ normal(0, log(100) / 2.57); // 0 < sigma_alpha1 < log(100)

  mu_alpha2 ~ normal(0, log(500) / 2.57); // -log(500) < mu_alpha1 < log(500)
  sigma_alpha2 ~ normal(0, log(50) / 2.57); // 0 < sigma_alpha1 < log(50)

  mu_beta1 ~ normal(0, 2 / 2.57); // -2 < mu_beta1 < 2
  sigma_beta1 ~ normal(0, 2 / 2.57); // 0 < sigma_beta1 < 2

  mu_beta2 ~ normal(0, 2 / 2.57); //-2 < mu_beta1 < 2
  sigma_beta2 ~ normal(0, 2 / 2.57); // 0 < sigma_beta1 < 2

  sigma ~ normal(0, log(100) / 2.57);
```

```
}

generated quantities {

  vector[N] logy_pred;

  for(i in 1:N){

    real mu = alpha_pop_sp[popid[i]]
    + beta_pop_sp[popid[i]] * (year[i] - 1980);
    logy_pred[i] = normal_rng(mu, sigma);

  }

}
```

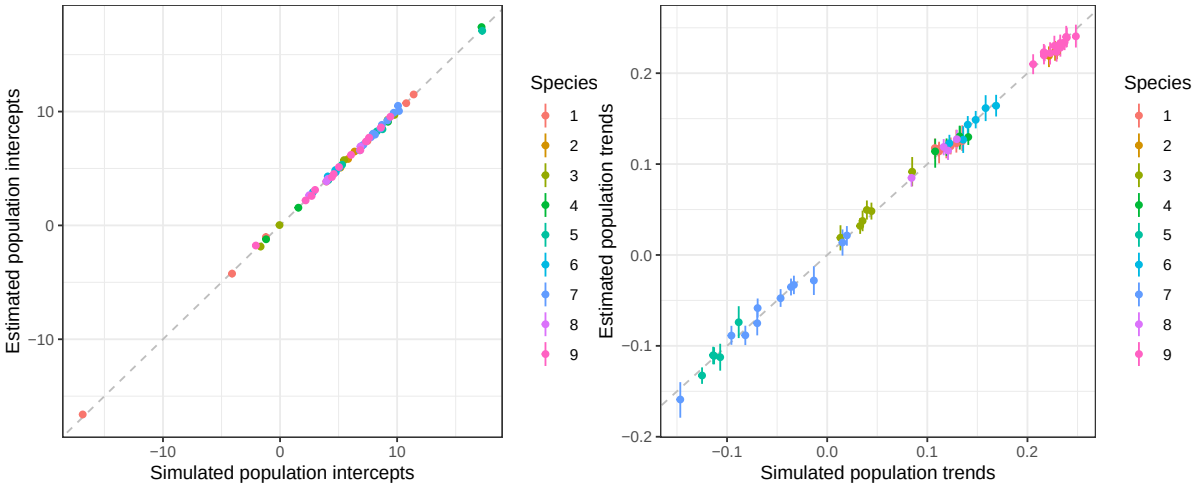
```
runmodel <- FALSE

data <- list(
  N = nrow(datasim),
  Nsp = nspecies,
  Npop = npops,
  y = y,
  year = years,
  spid_perpop = spid_perpop,
  popid = popid_perobs
)

if(runmodel){
  fit <- stan(file = "~/projects/forecastflows/analyses/stan/model1_nc2.stan",
              data = data,  iter=4000, warmup=3000, cores = 4, seed = 20012029)
  saveRDS(fit, file = file.path(wd, "output/fit/simulateddatafit.rds"))
}else{
  fit <- readRDS(file = file.path(wd, "output/fit/simulateddatafit.rds"))
}
```

Our first goal with simulated data is a simple check of our code (both the model code and simulated data code), given a high enough sample size relative to our effect sizes and error, the model should return our parameters. So our first step is to check this.

First, we start with the parameters, which include an overall estimate of the trend across species (this parameter would address our question that we decided on above). The red dashed line represents the true simulated value:



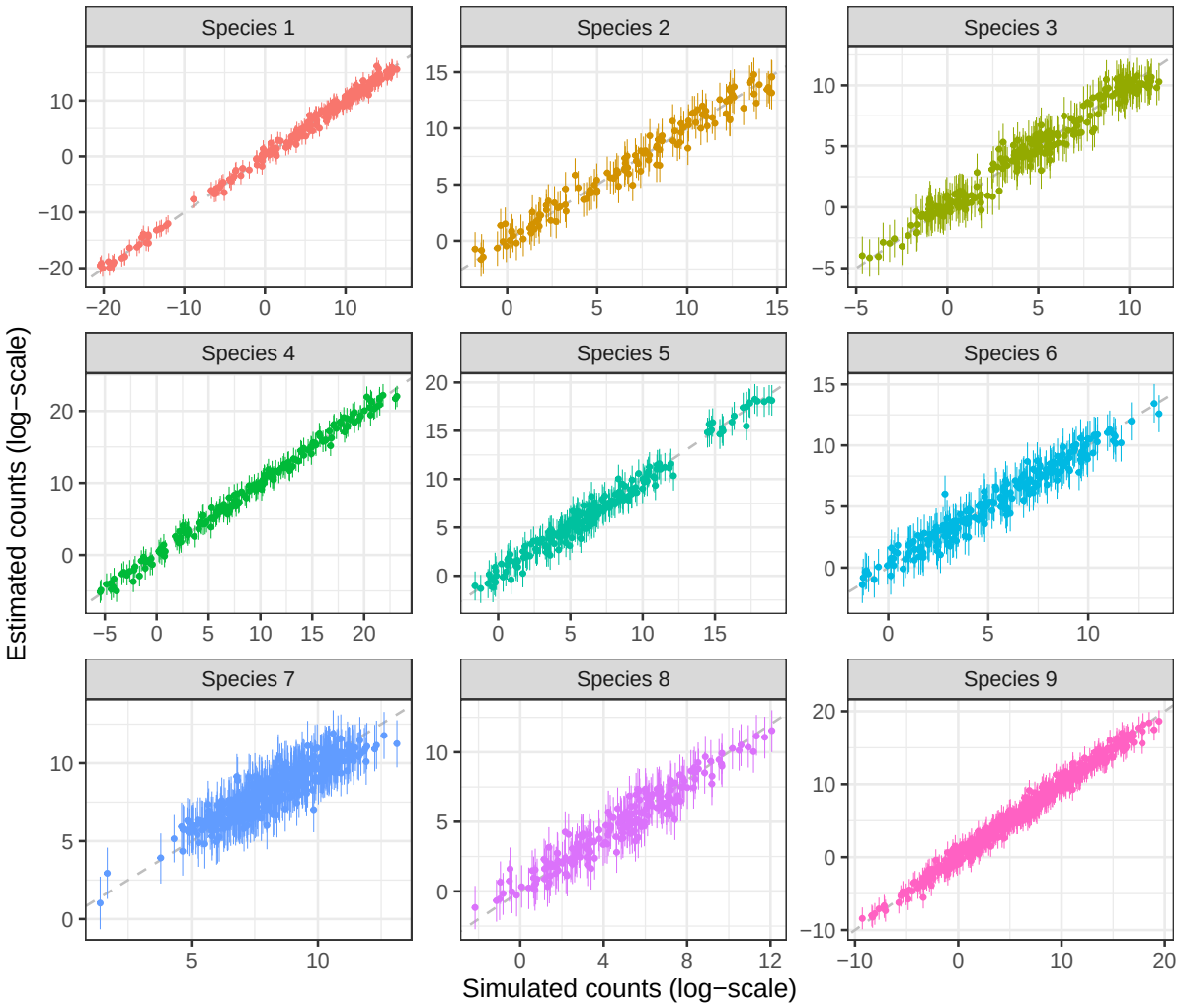
Now we check additional parameters. We find the model does a fair job at recovering population-level parameters.

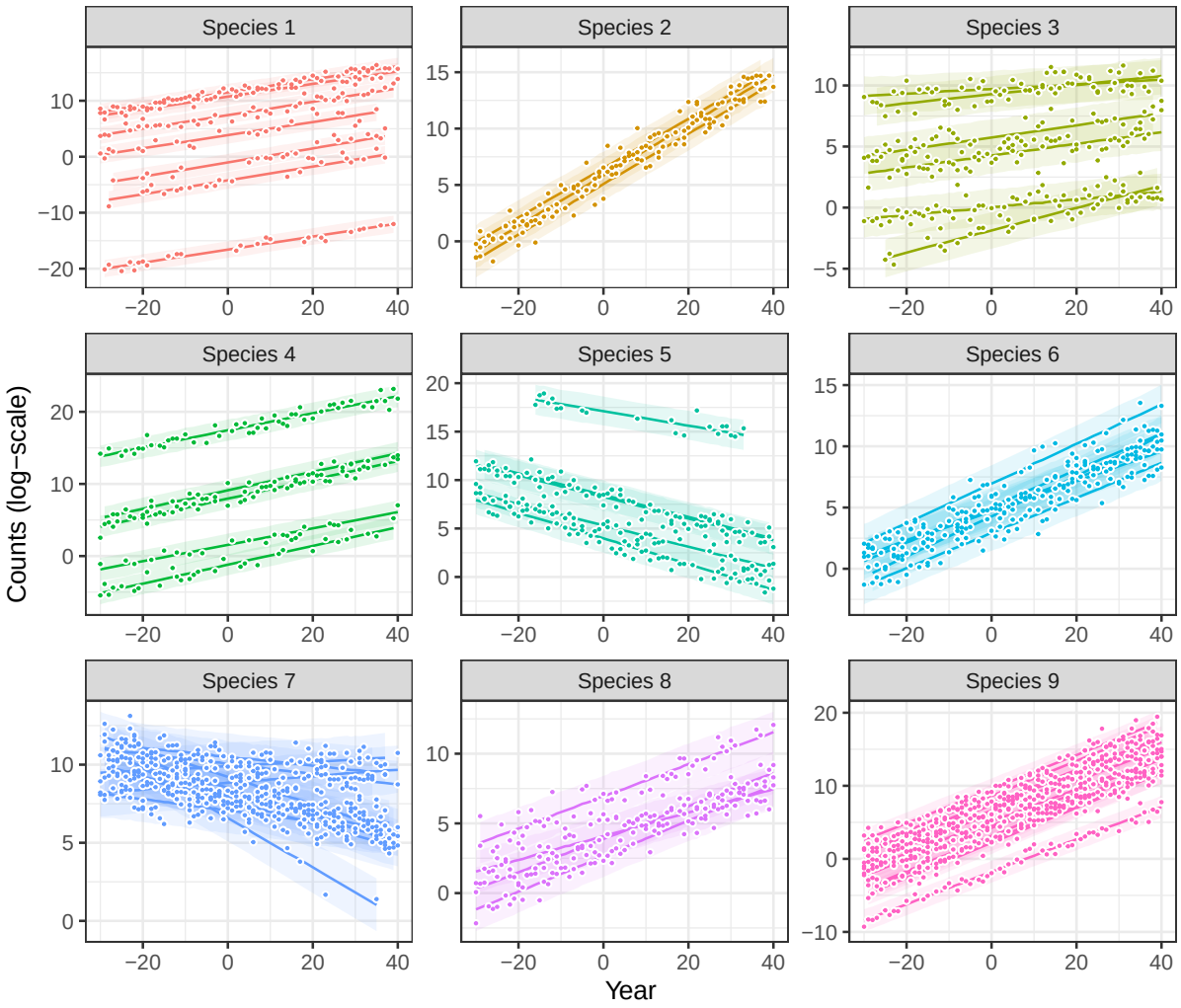


What if the model did not do a good job at recovering our parameters? Then we should likely adjust our sample size up and/or our error lower to make sure it is can (as a basic code test).

But what if the model can recover parameters only at sample sizes much higher than our data (or at error sizes much lower than our data)? Then we would need to take this important information forward in our analysis and be careful in how we interpret estimates from our model fit to empirical data (as we have a hint at this stage that getting robust estimates from our data and model together may be difficult, but we do not know the actual values of our parameters – including error and variance parameters – from our empirical data so we cannot be sure of whether we may have these issues yet).

For now, we also take a look at how model fits look against the simulated data (error bars and ribbon represent the 95% uncertainty intervals):


Species 1
Species 2
Species 3
Species 4
Species 5
Species 6
Species 7
Species 8
Species 9
Estimated counts (log−scale)
Simulated counts (log−scale)



These look pretty good, so we move onto the next step.

4 Step 4: Fit model to empirical data

Now that we have designed our question, built our model for it, and tested our model on simulated data, we can fit the model to our empirical data. We start by some basic clean-up of the dataframe so we have years in one column.

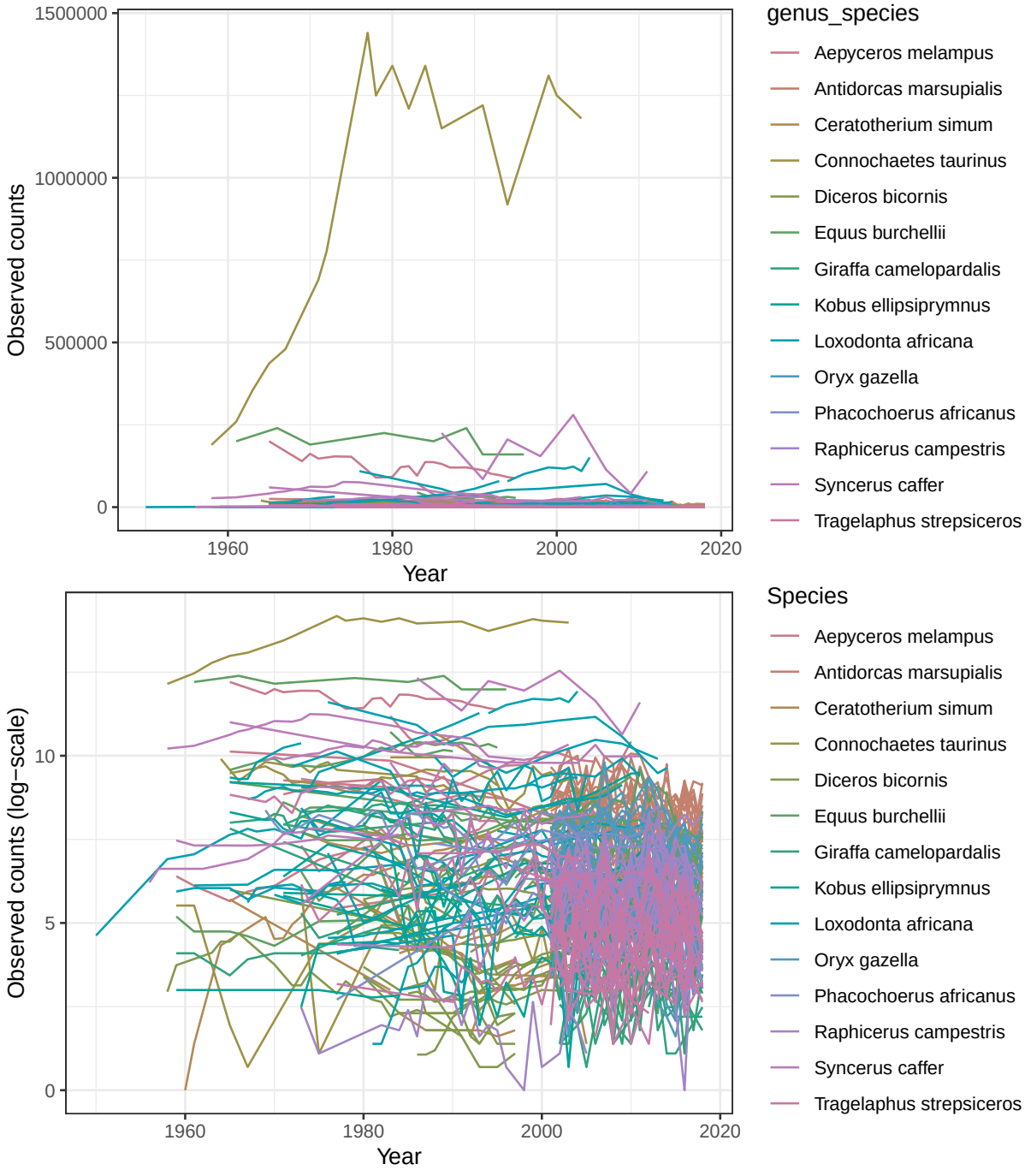
```
lpi_subset <- lpi_subset[, c("ID", "Genus", "Species", "Country", "Units",
                             paste0("X", 1950:2020))]
lpi_subset$genus_species <- paste(lpi_subset$Genus, lpi_subset$Species)
```

```
lpi_subset_rsh <-
  reshape(lpi_subset, direction = "long", idvar = "ID",
          varying =  paste0("X", 1950:2020), v.names = c("value"),
          timevar = "year", times = 1950:2020)
lpi_subset_rsh <- lpi_subset_rsh[lpi_subset_rsh$value != "NULL",]
lpi_subset_rsh$value <- as.numeric(lpi_subset_rsh$value)
```

The values in the database are not necessarily in the same units, so we subset to units related to absolute population sizes, which is what we have mostly been thinking of (some units are in density and we would need to build our model to incorporate those, but here we start first with a simpler model and can build up later). We take a quick look at the data to make sure it looks reasonable based on our domain knowledge.

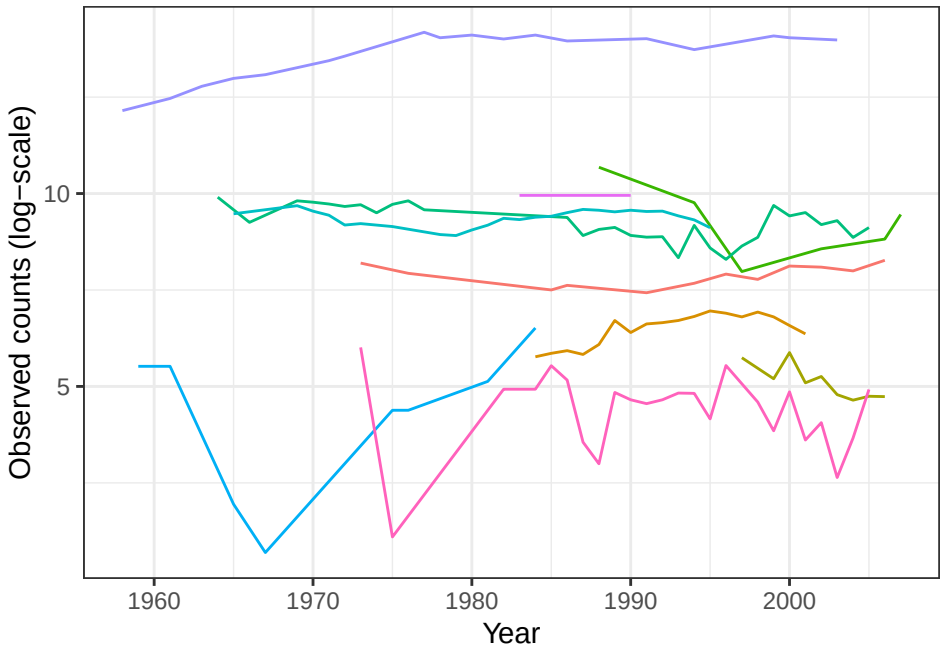
```
units_to_keep <- c("Estimate - entire population", "Estimate - entire population (August)",
                   "Individual", "individuals", "Individuals", "Number of adults",
                   "number of individuals", "Number of individuals", "Number of Individuals"
                   "Number of rhinos", "numbers of indviduals", "population estimate",
                   "Population estimate", "Population estimate (individuals)",
                   "Population estimates", "Total population size")

lpi_subset_rsh <- lpi_subset_rsh[lpi_subset_rsh$Units %in% units_to_keep,]

ggplot(data = lpi_subset_rsh) +
  geom_line(aes(x = year, y = value,
                group = paste0(genus_species, ID),
                color = genus_species)) +
  scale_color_manual(values = hcl.colors(14, palette = "Dark2"),
                     breaks = unique(lpi_subset_rsh$genus_species)[order(unique(lpi_subset_rs
  theme_bw() +
  labs(x = 'Year', y = 'Observed counts')
ggplot(data = lpi_subset_rsh[lpi_subset_rsh$value != 0,]) +
  geom_line(aes(x = year, y = log(value),
                group = paste0(genus_species, ID),
                color = genus_species)) +
  scale_color_manual(values = hcl.colors(14, palette = "Dark2"),
                     breaks = unique(lpi_subset_rsh$genus_species)[order(unique(lpi_subset_rs
  theme_bw() +
  labs(x = 'Year', y = 'Observed counts (log-scale)', color = "Species")
```

Now, we make a few final adjustments and fit the model:


1500000
1000000
500000
0
Observed counts
1960
1980
2000
2020
Year
genus_species
Aepyceros melampus
Antidorcas marsupialis
Ceratotherium simum
Connochaetes taurinus
Diceros bicornis
Equus burchellii
Giraffa camelopardalis
Kobus ellipsiprymnus
Loxodonta africana
Oryx gazella
Phacochoerus africanus
Raphicerus campestris
Syncerus caffer
Tragelaphus strepsiceros
10
5
0
Observed counts (log−scale)
1960
1980
2000
2020
Year
Species
Aepyceros melampus
Antidorcas marsupialis
Ceratotherium simum
Connochaetes taurinus
Diceros bicornis
Equus burchellii
Giraffa camelopardalis
Kobus ellipsiprymnus
Loxodonta africana
Oryx gazella
Phacochoerus africanus
Raphicerus campestris
Syncerus caffer
Tragelaphus strepsiceros

```
runmodel <- FALSE

# We now also find we have counts of 0, which is tricky for our approach
#(as we cannot evaluate log(0)), we would also flag this to return.
# As we iteratively improve the model, we would need to consider what
# processes generate these zeroes, and include them in our model.
lpi_subset_rsh <- lpi_subset_rsh[lpi_subset_rsh$value != 0,]

popids <- data.frame(ID = unique(lpi_subset_rsh$ID),
                     popid_num = 1:length(unique(lpi_subset_rsh$ID)))
spids <- data.frame(genus_species = unique(lpi_subset_rsh$genus_species),
                    spid_num = 1:length(unique(lpi_subset_rsh$genus_species)))

lpi_subset_rsh <- merge(lpi_subset_rsh, spids)
lpi_subset_rsh <- merge(lpi_subset_rsh, popids)

spid_perpop <- unique(lpi_subset_rsh[c("popid_num", "spid_num")])
spid_perpop <- spid_perpop[order(spid_perpop$popid_num),]

data <- list(
  N = nrow(lpi_subset_rsh),
  Nsp = max(spids$spid_num),
  Npop = max(popids$popid_num),
  y = lpi_subset_rsh$value,
  year = lpi_subset_rsh$year,
  spid_perpop = spid_perpop$spid_num,
  popid = lpi_subset_rsh$popid_num
)

if(runmodel){
  fit <- stan(file = "~/projects/forecastflows/analyses/stan/model1_nc2.stan",
              data = data,  iter=4000, warmup=3000, cores = 4, seed = 20012029)
  saveRDS(fit, file = file.path(wd, "output/fit/truedatafit.rds"))
}else{
  fit <- readRDS(file = file.path(wd, "output/fit/truedatafit.rds"))
}
```

Here we finally get to see how our model performs on the empirical data and are a step closer to answering our question, but we also start to notice some differences in our empirical data from our simulated data. In preparing our data for the model, we realize we have quite different amplitudes of variations across populations and species (on a log-scale), e.g. for blue wildebeest (each color represents a different population):

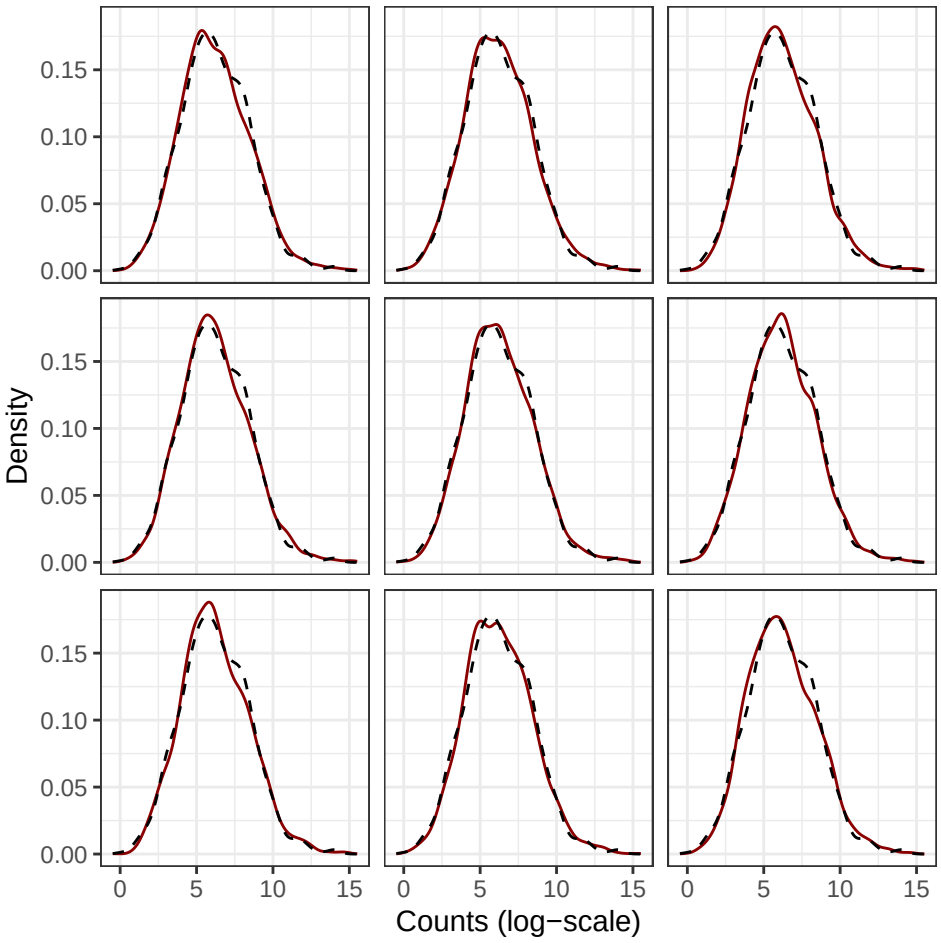


Our model may accommodate this, as we have allowed different populations to have different intercepts (starting population sizes), but the scale of this variation is much larger than we simulated so we would again want to flag this and check how our model performs with these data.

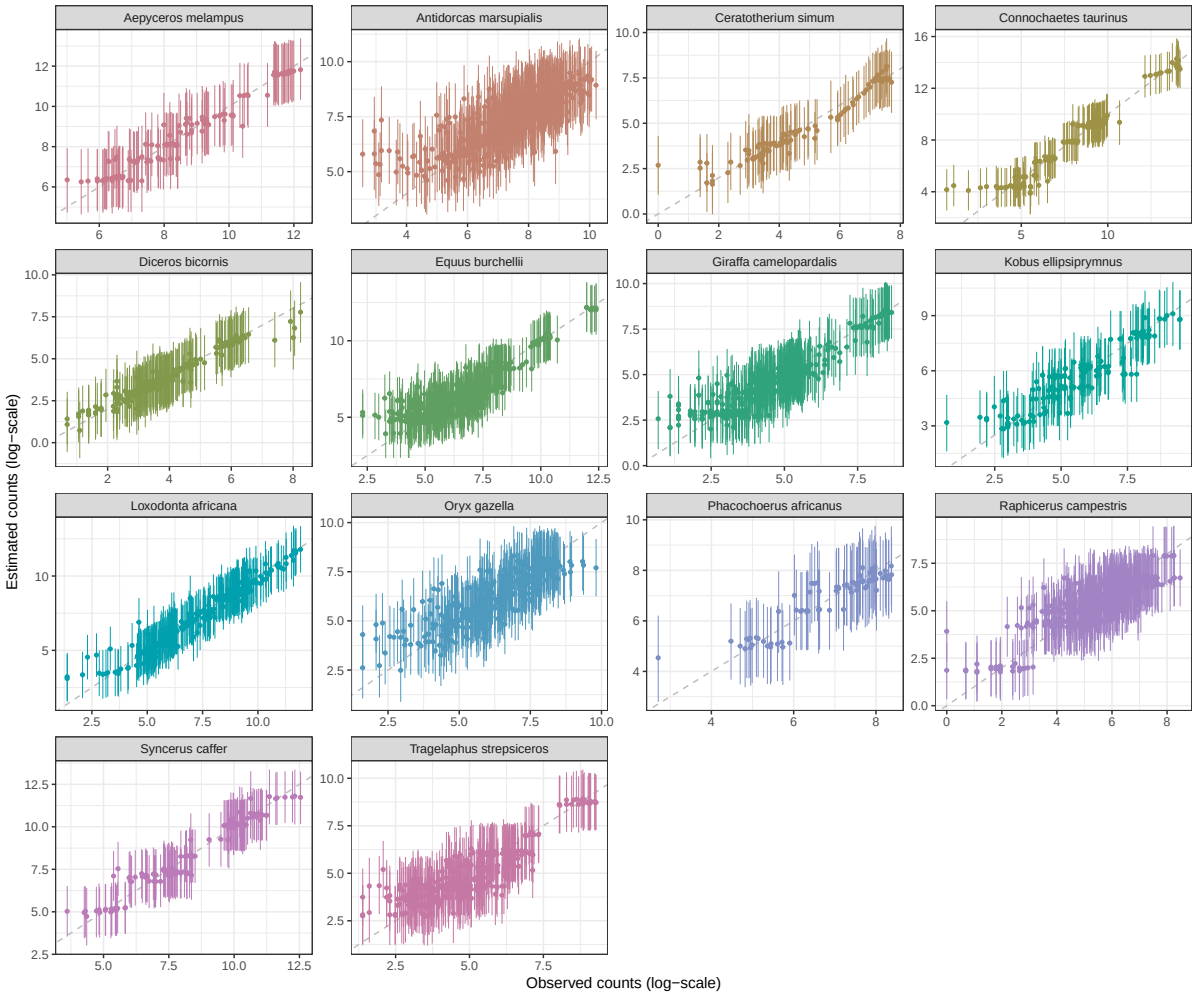
5 Step 5: Retrodictive and related checks

Now we use our fitted model parameters to generate predictions, and compare them to observations. This is similar to the simulating data step above, but differs in two major ways: (1) we simulate from the parameters estimated by the model fit to the empirical data (versus coming up with them based on our expert knowledge) and (2) we need to check across many predictions from the model to understand the full uncertainty of the model (and not just one draw from the model).

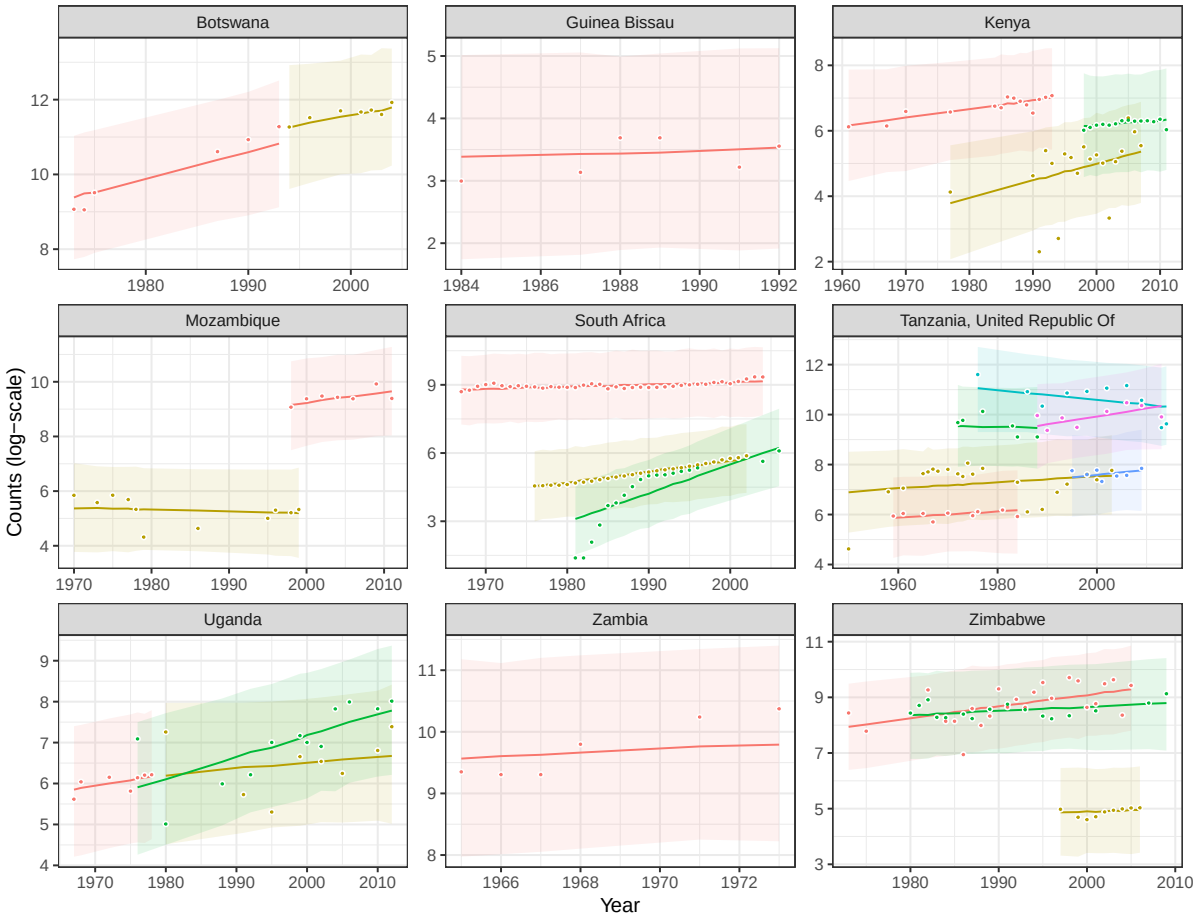
Here we start by visually comparing the 9 draws from the posterior predictive distribution of counts (across all species and populations, red line) to observed data (black dashed line):



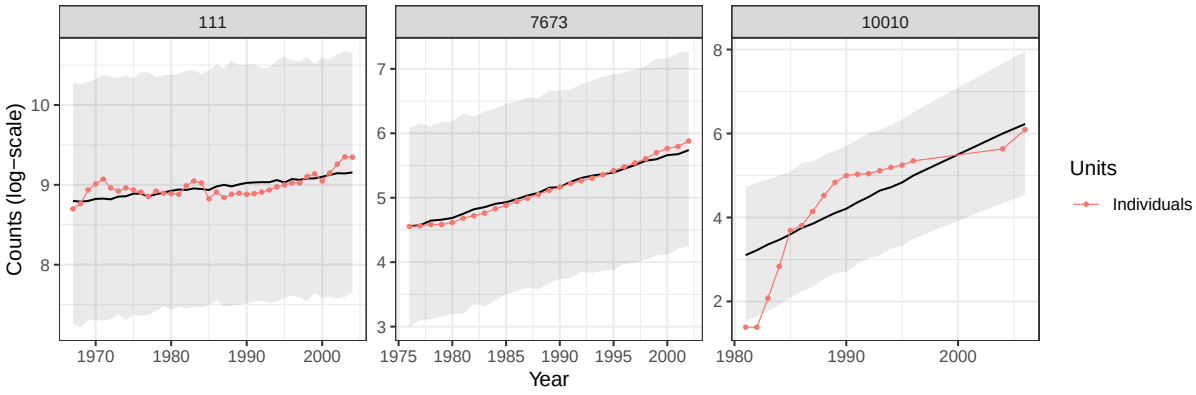
We then examine, by species, how well predictions from the model compare to the observed data; this is similar to many plots where we simply want to see how well the model seems to predict the data.



To evaluate our model more precisely, we need to check how the model predicts the data on finer scales. So we start by examining some species separately, e.g. here the African bush elephant (each color within a country represents a different population):



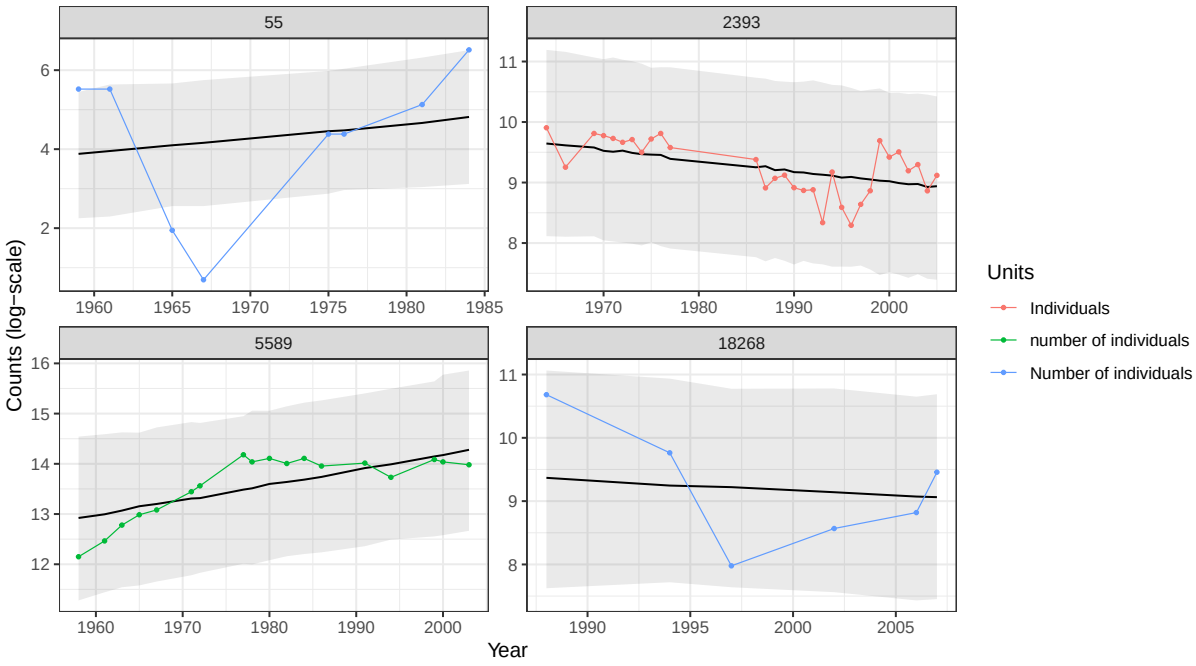
Focusing on populations in South Africa, we see some places where the linear model appears to fit reasonably well (ID 7673), but also one population where it does not appear as a very good fit (ID 10010).



The three populations are measured in the same units, but are very different in their sizes (ranging from a few hundred individuals to more than 10000), thus they likely represent

populations measured across different spatial scales.

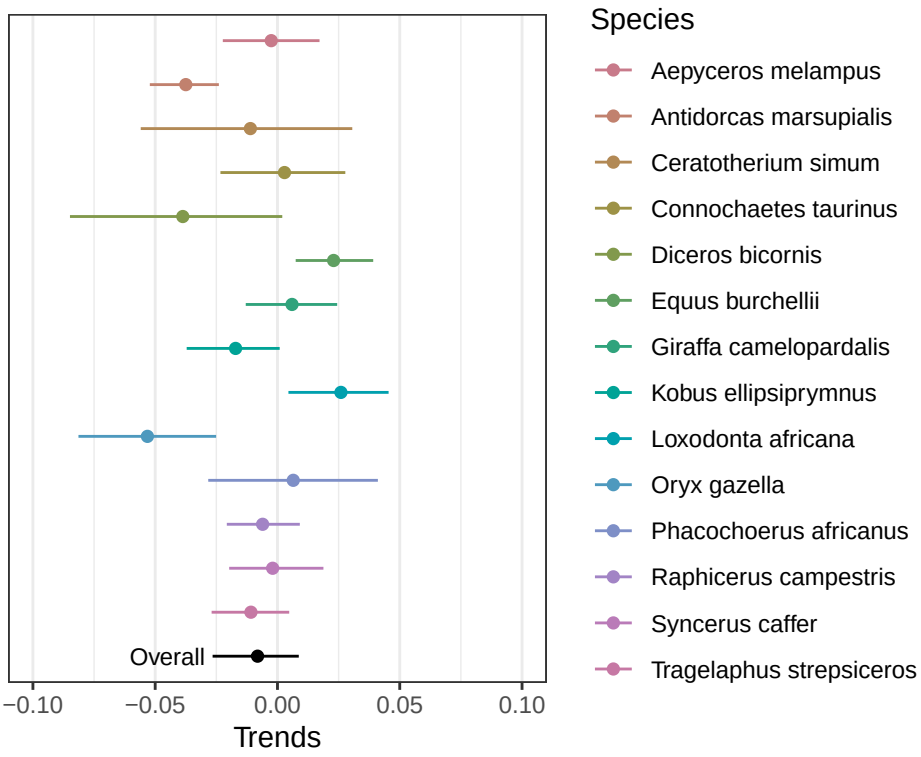
The same holds for other species and other countries, e.g. the blue wildebeest in Tanzania:



The four populations are also measured in the same basic units, but represent likely represent very different populations, spanning ones in small parks that cannot migrate to the large migrating herds (e.g., Serengeti Masai-Mara), though gathering additional metadata to fully check this would be another important component to add to our workflow for these data.

6 Inference

Once we are confident with our model (which is not truly the case here), we can look at the parameter values and check what the overall and species trends appear to be:

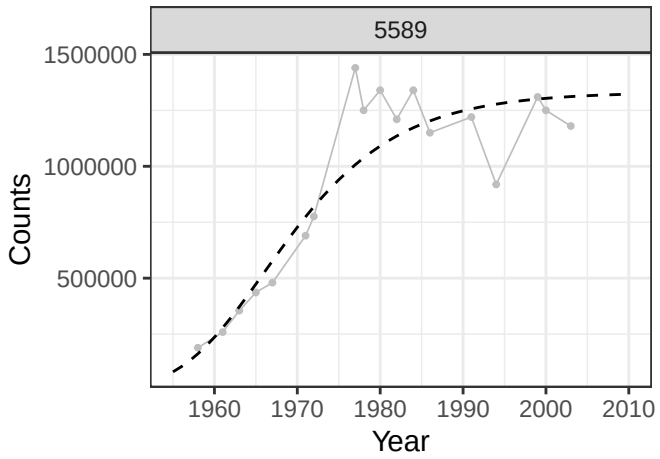


7 Feedbacks

At this point, we would have a lot of ideas of ways to improve this model that we would want to follow up on, including (but not limited to) questioning our assumption of one observational error σ for all populations, how we model population trends (e.g., there are many ways to model abundance), and (relatedly) our linearity assumption. One possibility would be to move to a non-linear regression, and use a more flexible model that would allow population numbers to stabilize, such as shown using a logistic model for one wildbeest population in Tanzania:

```
subset_wildebeest <- lpi_subset_rsh[lpi_subset_rsh$ID == 5589,]
subset_wildebeest$year <- subset_wildebeest$year-1980
gompodel <- nls(log(value) ~ SSlogis(year, Asym, b2, b3), data=subset_wildebeest)
logisfit <-data.frame(pred = predict(gompodel,
                                     newdata = data.frame(year = seq(-25,30,1))),
                      year = seq(-25,30,1))

ggplot(data = lpi_subset_rsh[lpi_subset_rsh$ID == 5589,]) +
  facet_wrap( ~ ID, scales = "free_y") +
  geom_point(aes(x = year, y = value),
```

```
            size = 0.7, color = "grey") +
  geom_line(aes(x = year, y = value),
            linewidth = 0.3, color = "grey") +
  geom_line(aes(x = year + 1980, y = exp(pred)),
           linewidth = 0.5, data = logisfit, linetype = 'dashed') +
  theme_bw() +
  labs(y = 'Counts', x = 'Year')
```



We also have seen variability in population trends within species (e.g., two apparently increasing populations and two apparently decreasing populations in Tanzania), so we would want to think a lot more about how we scale up from population to species to capture this variation in a way that is most useful to our question. For example, may want to think about adding country, which may capture differences in efforts related to conservation and thus predict trends (or we may have better metrics of what drives these trends, but given our global aim, we would ideally need the predictor for most/all populations) and we may want to think about how or if we partially pool species and populations within species (potentially using models other than a normal distribution for pooling). As we address these potential problems, we would likely begin to add in more countries, species, and populations.

We expect scaling up to more countries, species and populations may cause us to think carefully about the data we have in hand and our approach versus the question we aimed to answer. While many recent papers treat the LPI data as (effectively) a well stratified and randomized sample or even a good census, the LPI data is clearly neither. Thus, models that try to deal with such uneven sampling and build in more of the structure of data would be more appropriate. For example, most models for election forecasts now include regularization via partial pooling, post-stratification and other methods designed to adjust for the fact that many polls are 'non-representative.' Such non-representative polls may be very similar to LPI data. Our model includes some regularization via partial pooling, but we may need much more of an observational model added to our population demography model to accommodate the uneven

sampling. Estimates from such a model may be more uncertain, but likely also more correct given our combination of question and data.

This may lead us to realize we really don't have great data for this question and we would then need to prioritize better data collection efforts if we want good estimates to answer this question.

8 The full process

The small workflow we show here may seem like extra work to those who have not used it, and it is compared to many ecological workflows in practice. We argue that the extra effort is worth it given the critical questions we are addressing, and also that the process becomes much quicker with practice. The first time may be slow, but the resulting model will be better for it (even if it is still arguably simple, as the one we show here is), but the following uses of this iterative workflow will go faster and very likely yield improved models.