

Supporting documentation: Decreasing trends of chinstrap penguin breeding colonies in a region of major and ongoing environmental change suggest population level vulnerability: a reanalysis of Krüger (2023)

Chris Oosthuizen, Murray Christian, Azwianewi Makhado, Mzabalazo Ngwenya

2023-10-05

Contents

1	Krüger (2023) reanalysis	2
1.1	Load packages and set plotting theme	2
1.2	Load and process MAPPPD data for area 48.1:	3
1.3	Calculate the mean slope of the decrease per site (glm)	5
1.4	Some summary stats (Krüger 2023)	7
1.5	Identify first and last counts	7
1.6	Krüger 2023 Figure 2 proportion decrease	8
1.7	Corrected Figure 2	11
1.8	MCMCglmm mixed model data	13
1.9	Specify MCMCglmm mixed model prior (Krüger 2023)	14
1.10	Fit MCMCglmm mixed model (Krüger 2023)	14
1.11	MCMCglmm model checking	16
1.12	MCMCglmm Random effects (Krüger 2023)	19
1.13	Predicting counts from mixed model (1960 to 2020) (Krüger 2023)	20
1.14	How accurate are the predictions relative to observed data?	20
1.15	Figure 3A (Krüger 2023)	25
1.16	In the above plots, why are the predicted abundance so ‘wiggly’?	26
2	Change prediction to ‘confidence’ (credible) interval	27
2.1	Figure 3B (Krüger 2023): Latitude	37
2.2	Population change in 3 generations	38

3	Reanalysis for Oosthuizen et al.	44
3.1	Fit a better GLMM	44
3.2	MCMCglmm diagnostics for mc2	45
3.3	Predict using MCMCglmm mc2	48
3.4	Conditional model predictions	50
3.5	Marginal model predictions	56
3.6	Revised plots of latitude and slope (population change)	61
4	Oosthuizen et al data distribution figures	63
5	Oosthuizen et al - population change:	68
6	Predicting population change with entire posterior distribution	70

1 Krüger (2023) reanalysis

This script provides a reanalysis of Krüger (2023) (Citation: Krüger, L. (2023). Decreasing Trends of Chinstrap Penguin Breeding Colonies in a Region of Major and Ongoing Rapid Environmental Changes Suggest Population Level Vulnerability. Diversity, 15(3), 327.). The Krüger (2023) supplementary material provided reproducible R code for that study's analyses. We use that code here to replicate the original results. In addition, we provide additional analysis that cautions that the analysis performed by Krüger (2023) cannot support robust inference.

1.1 Load packages and set plotting theme

```
# Load packages
# data summary
library(reshape2)
library(plyr)
library(dplyr)
library(tidyverse)
#plots
library(ggplot2)
library(patchwork)
library(sjPlot)
#models
library(energy)
library(optimx)
library(minqa)
library(dfoptim)
library(MCMCglmm)

library(ggforce) # not part of original script, but needed to plot site trends below

# plot theme
```

```
th <- theme(axis.text=element_text(size=12, face="bold",colour="grey30"),
            axis.title=element_text(size=14,face="bold"),
            panel.grid.major = element_blank(),
            panel.grid.minor = element_blank(),
            title =element_text(size=12, face="bold",colour="black"))
```

1.2 Load and process MAPPPD data for area 48.1:

```
# Humphries et al. (2017) Mapping Application for Penguin Populations
# and Projected Dynamics (MAPPPD): data and tools for dynamic management
# and decision support. Polar Record 53 (269): 160-166 doi:10.1017/S0032247417000055
```

```
df <- read.csv(here::here("./data/mapppd AllCounts_V_4_0.csv"))
```

```
# subset chinstrap penguin
chins<-subset(df,common_name=="chinstrap penguin")
```

```
summary(as.factor(chins$common_name))
```

```
## chinstrap penguin
##           1342
```

```
summary(as.factor(chins$count_type))
```

```
## adults chicks  nests
##      91      147    1104
```

```
# use only nest counts
nests<-subset(chins,count_type=="nests")
```

```
# some populations had multiple counts over the same season:
# this summarises the count with the maximum nests
nestM<-ddply(na.omit(data.frame(nests)), c("season_starting","site_id"),
            summarise,
            nests=max(penguin_count),
            Lat=mean(latitude_epsg_4326),
            Lon=mean(longitude_epsg_4326))
```

Here, the na.omit function removes all rows where there are NA values (missing data). Some rows have missing information for: - the day of the count - the day and month of the count - the accuracy of the count - the vantage point (ground, boat, uav, vhr) - on 4 occasions there are no count data (NA). One can argue that counts with unknown accuracy, vantage point, or count dates should be excluded from analysis, as was done here. Alternatively, one can argue that it makes little sense to exclude counts (e.g., those with high accuracy) where the only data missing is the day on the month where the count was conducted. This is because we did not subset / select counts in any other way (e.g., data was not limited to 'accurate' counts, or those happening within a certain date limit). Thus, this paper could arguably have use more of the available count data (given what was used). It is also worth discussing whether counts with poor accuracy should have been included in analysis, and if included, what the impact of counts with poor accuracy can have on the results.

```
# summarizing number of populations and number of counts
countsN <-ddply(nestM, c("site_id", "Lat", "Lon"), summarise,
               ncounts=length(nests),
               interval=(max(season_starting)-min(season_starting)))
head(countsN)
```

```
##   site_id    Lat    Lon ncounts interval
## 1   ACUN -60.761 -44.637      1         0
## 2   AILS -60.780 -44.631      2        36
## 3   AITC -62.407 -59.752      4        21
## 4   AITK -60.738 -44.525      2        35
## 5   ALCO -64.240 -61.127      2        13
## 6   AMPH -60.684 -45.339      2        36
```

```
# summarizing number of populations and number of counts with more than 0 nests
countsN2<-ddply(subset(nestM,nests>0), c("site_id", "Lat", "Lon"), summarise,
                ncounts=length(nests),
                interval=(max(season_starting)-min(season_starting)))
head(countsN2)
```

```
##   site_id    Lat    Lon ncounts interval
## 1   ACUN -60.761 -44.637      1         0
## 2   AILS -60.780 -44.631      2        36
## 3   AITC -62.407 -59.752      4        21
## 4   AITK -60.738 -44.525      2        35
## 5   ALCO -64.240 -61.127      2        13
## 6   AMPH -60.684 -45.339      2        36
```

```
summary(as.factor(countsN2$ncounts))
```

```
##   1  2  3  4  5  6  7  8  9 10 11 12 14 15 21
## 148 89 14 8  4  2  2  3  1  3  2  1  1  1  2
```

```
npops=length(countsN2$ncounts[countsN2$ncounts>1])
npops # number of populations
```

```
## [1] 133
```

```
nestM2<-merge(nestM, countsN) # number of counts for each population by merging
```

```
# test for Poisson distribution (Poisson M-test)
poisson.mtest(nestM2$nests[nestM2$ncounts>1 & nestM2$nests>0], R=199)
```

```
##
## Poisson M-test
##
## data: nestM2$nests[nestM2$ncounts > 1 & nestM2$nests > 0] replicates: 199
## M-CvM = 158.43, p-value = 0.1608
## sample estimates:
## [1] 3006.691
```

Here, the `poisson.mtest` is conducted on all the data (`nestM2$ nests[nestM2$ncounts>1 & nestM2$ncounts>0]`). Yet, a glm is run per site. Should this test not be conducted at the site level, if we are conducting site-specific analysis? Tests for a Poisson distribution at the site level is not really possible as most sites only have two counts. Regardless, we can probably just assume a Poisson distribution because counts are often Poisson distributed.

1.3 Calculate the mean slope of the decrease per site (glm)

```
# subset only populations with at least 2 counts and with any nest recorded
nestm3 <-subset(nestM2,ncounts>1 & nests>0)

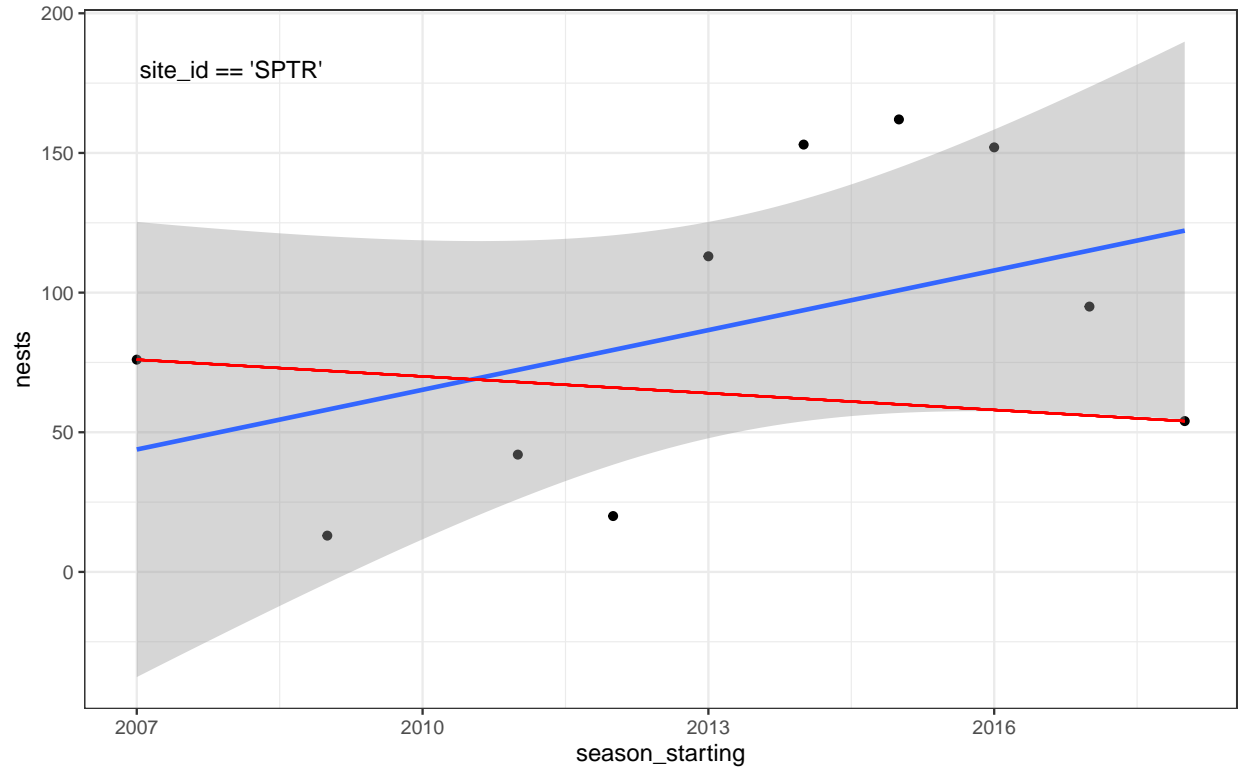
# calculating population level slope
slopeN <-na.omit(data.frame((nestm3 %>%
                             group_by(site_id,Lat,Lon) %>%
                             do({
                               mod=glm(nests~season_starting,family="poisson",
                                       data=.)
                               data.frame(Intercept=coef(mod)[1],
                                           Slope=coef(mod)[2])))))
```

1.3.1 Sanity check:

We use ‘sanity check’ exploratory plots as part of additional data exploration

```
# WCO: aside some sites have positive glm slopes (growth) but
# negative proportion change (calculated below) e.g. site_id == "SPTR"
# Reminds me of recommendation by Hill et al. (2019) J of Crustacean Biology 39:316-322.
# "Avoid diagnosing or rejecting a multi-year trend based on a comparison of two years."

chck = subset(nestm3, nestm3$site_id == "SPTR")
chck = chck[order(chck$season_starting),]
ggplot(data = chck, aes(season_starting, nests)) +
  geom_point()+
  geom_smooth(method='lm', formula= y~x) +
  geom_segment(aes(x = min(chck$season_starting),
                    y = chck$nests[1],
                    xend = max(chck$season_starting),
                    yend = chck$nests[10]), colour = "red") +
  theme_bw()+
  annotate("text", x=2008, y=180, label= "site_id == 'SPTR'")
```



*# The model slopes are the same if the decrease is the same.
 # E.g. these two sites halved in size and have the same slope (0.01925409)
 # (but different intercepts)*

```
subset(nestm3, nestm3$site_id == 'ANDE')
```

##	site_id	Lat	Lon	season_starting	nests	ncounts	interval
## 14	ANDE	-60.757	-44.601	2019	200	2	36
## 15	ANDE	-60.757	-44.601	1983	100	2	36

```
subset(slopeN, slopeN$site_id == 'ANDE')
```

##	site_id	Lat	Lon	Intercept	Slope
## 6	ANDE	-60.757	-44.601	-33.57569	0.01925409

```
subset(nestm3, nestm3$site_id == 'AILS')
```

##	site_id	Lat	Lon	season_starting	nests	ncounts	interval
## 2	AILS	-60.78	-44.631	2019	3000	2	36
## 3	AILS	-60.78	-44.631	1983	6000	2	36

```
subset(slopeN, slopeN$site_id == 'AILS')
```

##	site_id	Lat	Lon	Intercept	Slope
## 1	AILS	-60.78	-44.631	46.88037	-0.01925409

Here it is clear that there is rounding of numbers (100, 200, 3000, 6000). Rounding can contribute to uncertainty in true trends.

1.4 Some summary stats (Krüger 2023)

```
sloN <-merge(slopeN,countsN2) # number of counts for each population by merging
summary(as.factor(sloN$ncounts))
```

```
##  2  3  4  5  6  7  8  9 10 11 12 14 15 21
## 89 14  8  4  2  2  3  1  3  2  1  1  1  2
```

```
sloN$stdSlope<-sloN$Slope/sloN$interval
mean(sloN$Slope)
```

```
## [1] -0.02045084
```

```
sd(sloN$Slope)/sqrt(length(sloN$Slope)-1)
```

```
## [1] 0.007251265
```

```
# Note: original code was < but need to include <= to get the same results as older R versions (difference)
mean(sloN$Slope[sloN$Slope<=0])
```

```
## [1] -0.04960635
```

```
# Note: original code was < but need to include <= to get the same results as older R versions (difference)
sd(sloN$Slope[sloN$Slope<=0])/sqrt(length(sloN$Slope[sloN$Slope<=0])-1)
```

```
## [1] 0.009966612
```

```
# number of populations
length(sloN$Slope)
```

```
## [1] 133
```

```
# Note: number of decreasing populations: # original code was < but need to include <= to get the same results
length(sloN$Slope[sloN$Slope<=0])
```

```
## [1] 83
```

```
# proportion of decreasing populations
length(sloN$Slope[sloN$Slope<=0])/length(sloN$Slope)
```

```
## [1] 0.6240602
```

1.5 Identify first and last counts

```
# identify year of first count
firstN<-ddply(nestM, c("site_id"), summarise,
             Ncounts=length(nests),
             season_starting=min(season_starting))
```

```
# counts on the first year
firstCount<-merge(nestM,firstN)
```

```
# identify year of last count
lastN<-ddply(nestM, c("site_id"), summarise,
             season_starting=max(season_starting))
```

```
# counts of the last year
lastCount<-merge(nestM,lastN)
```

```
summary(firstCount$Ncounts)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.000   1.000   2.000   2.278   2.000   21.000
```

```
# change names to join data frames
names(firstCount)[names(firstCount) == 'season_starting'] <- 'First'
names(firstCount)[names(firstCount) == 'nests'] <- 'FirstCount'
names(lastCount)[names(lastCount) == 'season_starting'] <- 'Last'
names(lastCount)[names(lastCount) == 'nests'] <- 'LastCount'
firlas<-merge(firstCount,lastCount,by=c("site_id","Lat","Lon")) # first and last counts
firlas<-subset(firlas,Ncounts>1) # subset only pops with more than one count
firlas$PercChange<-((firlas$LastCount/firlas$FirstCount)-1)*100 #percentual change
firlas$PercChange[is.na(as.numeric(firlas$PercChange))]<-0 # make NA = 0
Slope.Counts<-merge(firlas,sloN,by=c("site_id","Lat","Lon")) # merge slope and counts
summary(Slope.Counts$PercChange) ##### percent change at the population level
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     -100.00  -61.62  -23.08   11.31   28.33   900.00
```

```
sd(Slope.Counts$PercChange) /sqrt(length(Slope.Counts$PercChange)-1) # standard error
```

```
## [1] 11.2554
```

1.6 Krüger 2023 Figure 2 proportion decrease

```
#subset only decreasing populations (WCO: THIS ALSO SELECTS (ONE) STABLE POPULATION)
# original code was < but need to include <= to get the same results as older R versions (different tre
```

```
decr<-subset(Slope.Counts,Slope<=0)
decr$YearDecr<-(-1*decr$Slope) # decrease per year
decr$PercDecr<-(-1*decr$PercChange) # absolute percent decrease
```

```
### classify range of decrease in categories ($decrCat)
```



```

decr$decrCat[decr$PercDecr<=25]<-"less than 25%"
decr$decrCat[decr$PercDecr>25 & decr$PercDecr<=50]<-"25% to 50%"
decr$decrCat[decr$PercDecr>50 & decr$PercDecr<=75]<-"50% to 75%"
# WCO: CODING ERROR / BUG. SELECTS >55 %, NOT 75%
decr$decrCat[decr$PercDecr>55]<-"more than 75%"
decr$decrCat<-factor(decr$decrCat,levels=c("less than 25%",
                                           "25% to 50%",
                                           "50% to 75%",
                                           "more than 75%")) # order of levels

n<-ddply(decr, c("decrCat"), summarise,
         N=length(FirstCount))
n

```

```

##      decrCat  N
## 1 less than 25% 19
## 2   25% to 50% 21
## 3   50% to 75%  5
## 4 more than 75% 38

```

```

sum(n$N) # check number of pops

```

```

## [1] 83

```

```

n$perc<-n$N/83 # percentage of populations in each categories

```

```

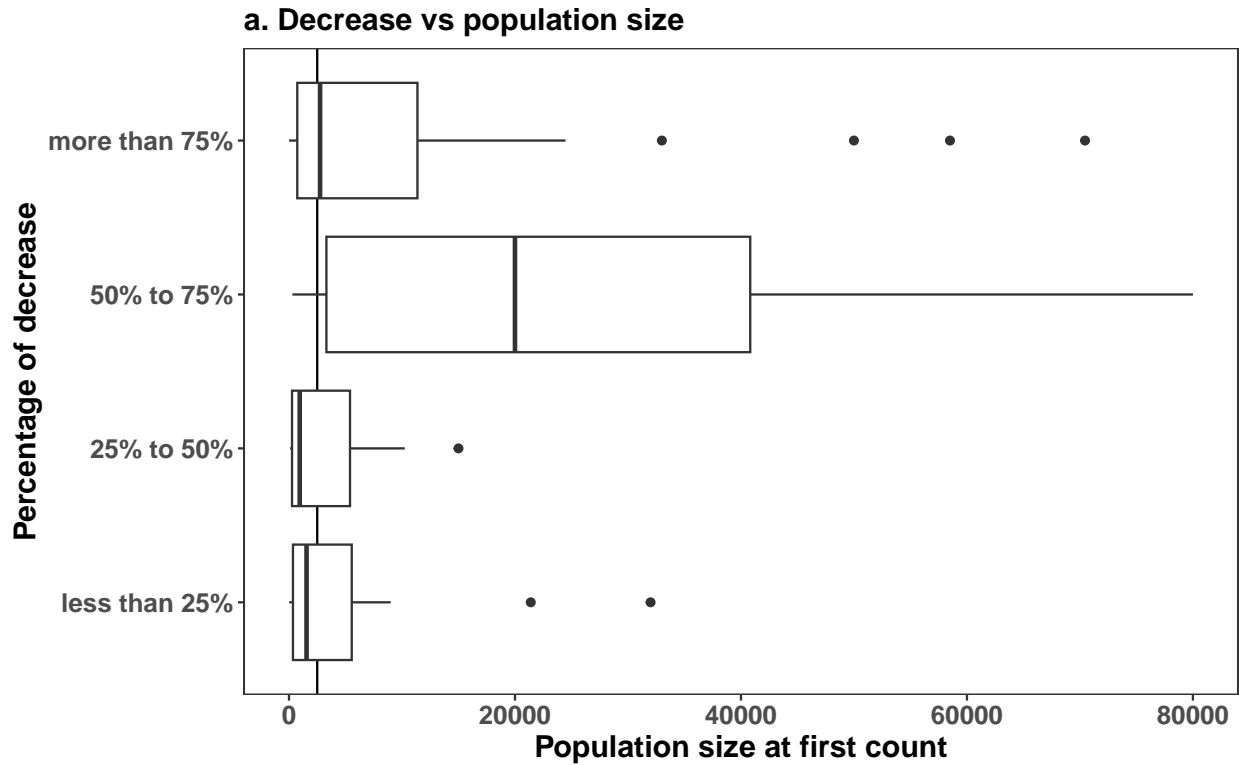
perc_original = n$perc

```

```

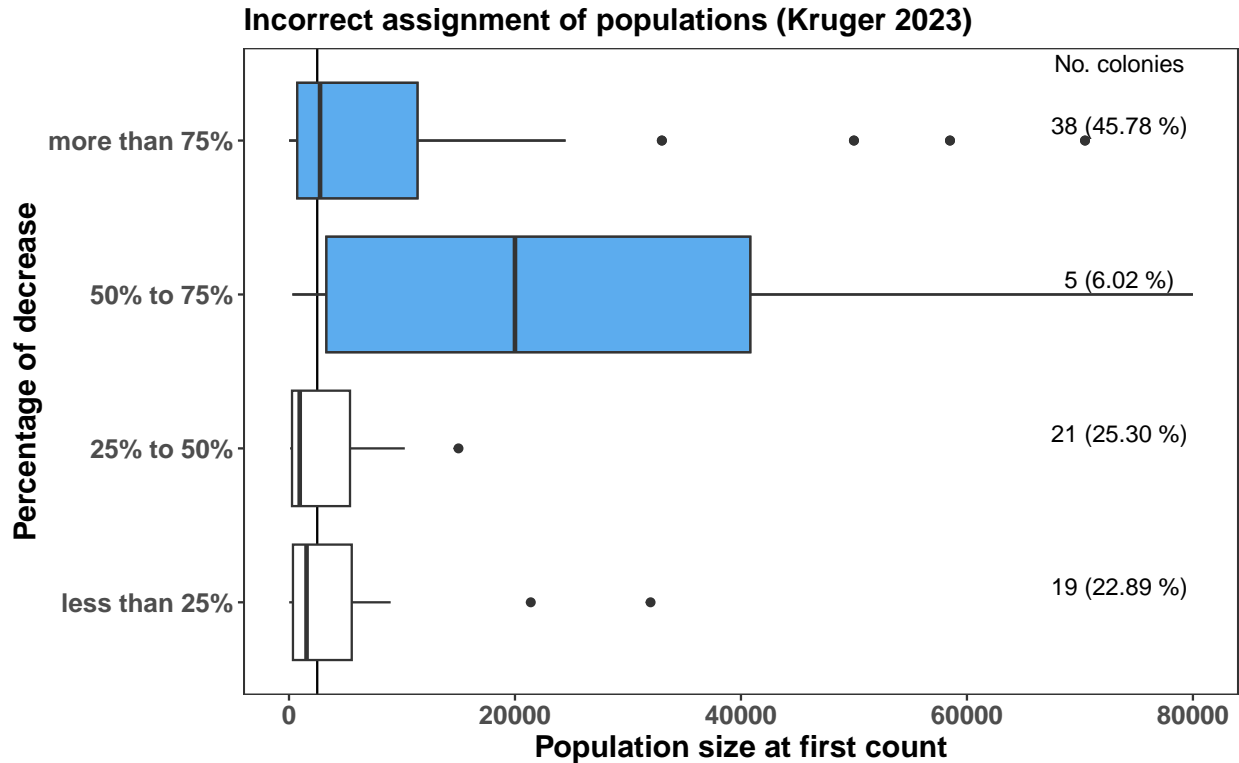
#figure 2
ggplot(decr,aes(decrCat,FirstCount))+
  geom_hline(yintercept=2500)+
  geom_boxplot()+
  coord_flip()+theme_bw()+th+
  xlab("Percentage of decrease")+
  ylab("Population size at first count")+
  ggtitle(label="a. Decrease vs population size")

```



```
fig2a = ggplot(decr,aes(decrCat,FirstCount))+
  geom_hline(yintercept=2500)+
  geom_boxplot()+
  geom_boxplot(data=decr[decr$decrCat=="more than 75%",],
               aes(x = decrCat, y = FirstCount),fill="steelblue2")+
  geom_boxplot(data=decr[decr$decrCat=="50% to 75%",],
               aes(x = decrCat, y = FirstCount),fill="steelblue2")+
  coord_flip()+theme_bw()+th+
  xlab("Percentage of decrease")+
  ylab("Population size at first count")+
  ggtitle(label="Incorrect assignment of populations (Kruger 2023)") +
  annotate("text", x = c(1.1, 2.1, 3.1, 4.1, 4.5),
           y = c(73500, 73500, 73500,73500,73500),
           label = c("19 (22.89 %)", "21 (25.30 %)",
                     "5 (6.02 %)", "38 (45.78 %)",
                     "No. colonies"), size=4)
```

fig2a



The above figure is incorrect as it includes population declines >55 % in the >75 % category (coding error). In other words, there are too many populations included in the >75 % category.

1.7 Corrected Figure 2

```
# subset only decreasing populations (THIS ALSO SELECTS 1 STABLE POPULATION)
decr<-subset(Slope.Counts,Slope<0)
decr$YearDecr<-(-1*decr$Slope) # decrease per year
decr$PercDecr<-(-1*decr$PercChange) # absolute percent decrease
### classify range of decrease in categories ($decrCat)
decr$decrCat[decr$PercDecr<=25]<-"less than 25%"
decr$decrCat[decr$PercDecr>25 & decr$PercDecr<=50]<-"25% to 50%"
decr$decrCat[decr$PercDecr>50 & decr$PercDecr<=75]<-"50% to 75%"
# This line had a CODING ERROR / BUG. it selected >55 %, NOT 75%
decr$decrCat[decr$PercDecr>75]<-"more than 75%"
decr$decrCat<-factor(decr$decrCat,levels=c("less than 25%",
      "25% to 50%",
      "50% to 75%",
      "more than 75%")) # order of levels

n<-ddply(decr, c("decrCat"), summarise,
  N=length(FirstCount))
n
```

```
##      decrCat  N
## 1 less than 25% 18
```

```
## 2    25% to 50% 21
## 3    50% to 75% 26
## 4 more than 75% 17
```

```
#sum(n$N) # check number of pops

n$perc<-n$N/83 # percentage of populations in each categories

perc_corrected =n$perc

## original manuscript percentage of populations in each category
print(perc_original)
```

```
## [1] 0.22891566 0.25301205 0.06024096 0.45783133
```

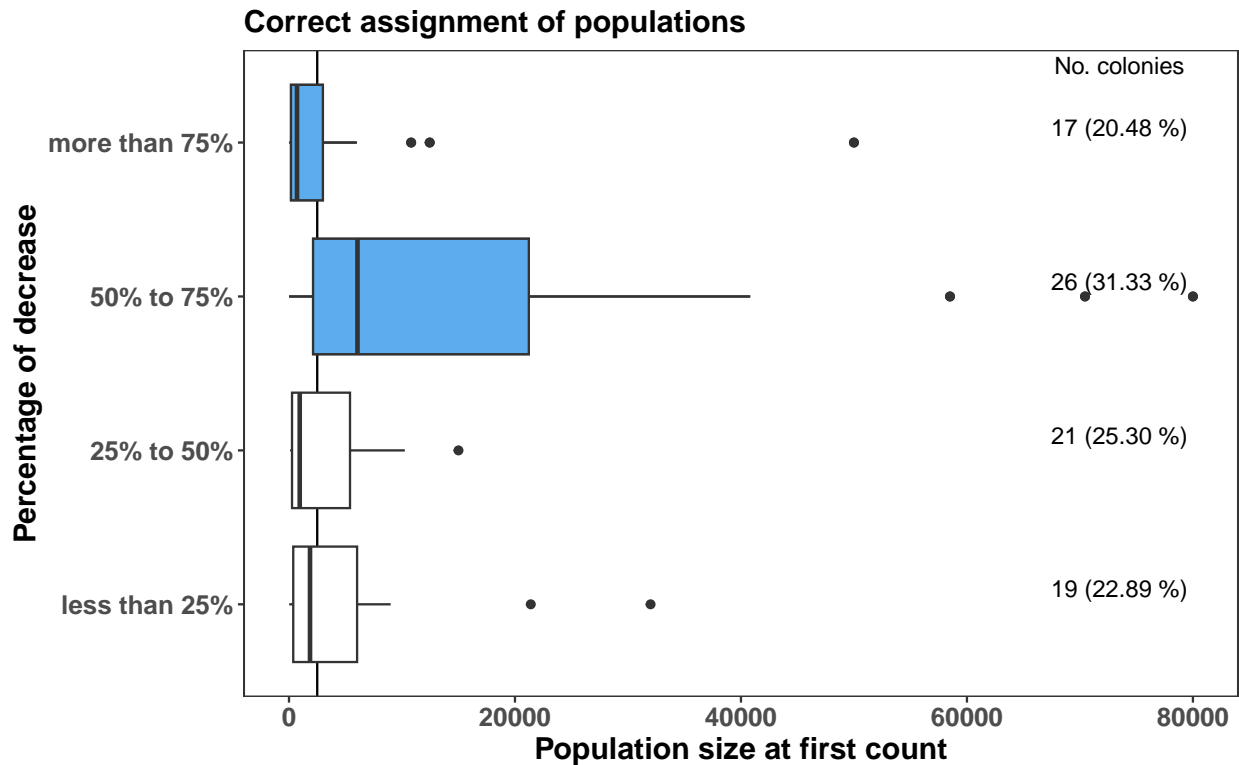
```
#### corrected percentage of populations in each category
perc_corrected
```

```
## [1] 0.2168675 0.2530120 0.3132530 0.2048193
```

```
#figure 2 corrected

fig2b = ggplot(decr,aes(decrCat,FirstCount))+
  geom_hline(yintercept=2500)+
  geom_boxplot()+
  geom_boxplot(data=decr[decr$decrCat=="more than 75%",],
               aes(x = decrCat, y = FirstCount),fill="steelblue2")+
  geom_boxplot(data=decr[decr$decrCat=="50% to 75%",],
               aes(x = decrCat, y = FirstCount),fill="steelblue2")+
  coord_flip()+theme_bw()+th+
  xlab("Percentage of decrease")+
  ylab("Population size at first count")+
  ggtitle(label="Correct assignment of populations") +
  annotate("text", x = c(1.1, 2.1, 3.1, 4.1, 4.5),
             y = c(73500, 73500, 73500,73500,73500),
             label = c("19 (22.89 %)", "21 (25.30 %)",
                       "26 (31.33 %)", "17 (20.48 %)", "No. colonies"),
             size=4)

fig2b
```



```
# library(cowplot)
# plot_grid(fig2a, fig2b)

## Save Plot
# pdf("./Figure 2.pdf", useDingbats = FALSE, width = 14, height = 7)
# plot_grid(fig2a, fig2b,
#           labels = "AUTO", scale = 0.9, vjust = 2, hjust = -4)
# dev.off()
```

1.8 MCMCglmm mixed model data

```
nestM3 <- nestm3 #populations with at least 2 counts and with any nest recorded
length(unique(nestM3$site_id))
```

```
## [1] 146
```

What is the sample size per site? Krüger (2023) reports 133 sites, but the count here is 146 unique sites. But the code gave 133 sites above. Why do we get both 133 and 146? Some sites have 2 counts (e.g., TAYL) but one of the counts are zero. TAYL is included in the nestm3 data frame. It remains in there because the filter (nestm3; paper's code above) is on `ncounts > 1 & nests > 0`, and `nests` is the variable that counts how many nest counts were made. But that variable does not condition on the counts being more than 0. 133 sites had more than one data point in nestM3. 146 sites were included in the GLMM analysis, including some sites with only 1 count.

1.9 Specify MCMCglmm mixed model prior (Krüger 2023)

```
prior<- list(R = list(V = 1, nu = 0.002),
             G = list(G1 = list(V = diag(2), nu = 0.002,
                                alpha.mu = rep(0, 2),
                                alpha.V= diag(133, 2, 2))))
```

1.10 Fit MCMCglmm mixed model (Krüger 2023)

```
mc1<-MCMCglmm(nests~season_starting, random=~us(1 + Lat):site_id, rcov=~units,
              family="poisson", mev=NULL,
              data=nestM3, start=NULL, nodes="ALL", scale=TRUE,
              nitt=13000, thin=10, burnin=3000,
              pr=T, pl=FALSE, verbose=TRUE, DIC=TRUE, singular.ok=FALSE,
              saveX=TRUE, prior=prior, saveZ=TRUE, saveXL=TRUE, slice=FALSE,
              ginverse=NULL, trunc=FALSE)
```

```
##
##                               MCMC iteration = 0
##
## Acceptance ratio for liability set 1 = 0.000443
##
##                               MCMC iteration = 1000
##
## Acceptance ratio for liability set 1 = 0.222374
##
##                               MCMC iteration = 2000
##
## Acceptance ratio for liability set 1 = 0.287476
##
##                               MCMC iteration = 3000
##
## Acceptance ratio for liability set 1 = 0.309278
##
##                               MCMC iteration = 4000
##
## Acceptance ratio for liability set 1 = 0.315363
##
##                               MCMC iteration = 5000
##
## Acceptance ratio for liability set 1 = 0.314639
##
##                               MCMC iteration = 6000
##
## Acceptance ratio for liability set 1 = 0.314772
##
##                               MCMC iteration = 7000
##
## Acceptance ratio for liability set 1 = 0.315119
##
```

```
## MCMC iteration = 8000
##
## Acceptance ratio for liability set 1 = 0.314848
##
## MCMC iteration = 9000
##
## Acceptance ratio for liability set 1 = 0.315489
##
## MCMC iteration = 10000
##
## Acceptance ratio for liability set 1 = 0.314858
##
## MCMC iteration = 11000
##
## Acceptance ratio for liability set 1 = 0.316370
##
## MCMC iteration = 12000
##
## Acceptance ratio for liability set 1 = 0.315242
##
## MCMC iteration = 13000
##
## Acceptance ratio for liability set 1 = 0.315392
```

*# Note: low ESS for random effects. Random effect ESS was 25-27 in Kruger (2023),
(see supplement), so this is not only a problem that we are encountering.*

```
summary(mcl)
```

```
##
## Iterations = 3001:12991
## Thinning interval = 10
## Sample size = 1000
##
## DIC: 4785.787
##
## G-structure: ~us(1 + Lat):site_id
##
## post.mean 1-95% CI u-95% CI eff.samp
## (Intercept):(Intercept).site_id 1991.5479 28.490585 4546.266 71.00
## Lat:(Intercept).site_id 32.4988 0.255875 73.122 71.03
## (Intercept):Lat.site_id 32.4988 0.255875 73.122 71.03
## Lat:Lat.site_id 0.5313 0.006634 1.185 71.12
##
## R-structure: ~units
##
## post.mean 1-95% CI u-95% CI eff.samp
## units 0.2941 0.2539 0.3441 1000
##
## Location effects: nests ~ season_starting
##
## post.mean 1-95% CI u-95% CI eff.samp pMCMC
## (Intercept) 27.743045 20.730480 34.391312 824.5 <0.001 ***
## season_starting -0.010443 -0.013982 -0.007111 823.3 <0.001 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Random effect syntax: ~us fit different variances across each component in formula, plus the co-variances. The linear model inside the variance function has two parameters, an intercept(1) and a regression slope associated with latitude. Each site now has an intercept and a slope specified. But slope (latitude) does not vary within site, and there is only one count per year, per site. This is not a good random effect model structure.

1.10.1 Sanity check:

```
# Each site only has one latitude value (should be 1)
uniqueLat = nestM3 %>%
  group_by(site_id) %>%
  dplyr::summarise(count = n_distinct(Lat))

max(uniqueLat$count)
```

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

```
length(unique(nestM3$Lat))
```

```
## [1] 140
```

```
length(unique(nestM3$site_id))
```

```
## [1] 146
```

1.11 MCMCglmm model checking

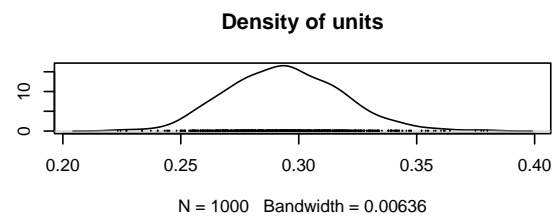
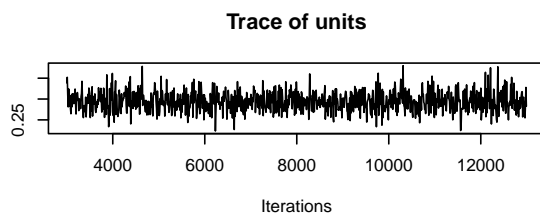
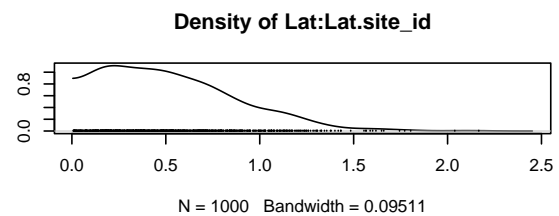
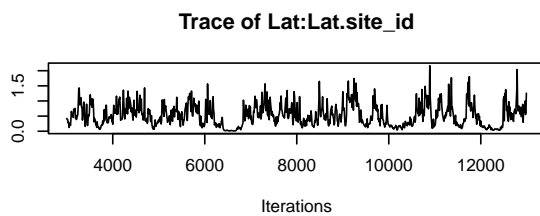
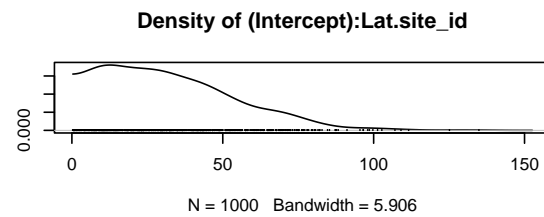
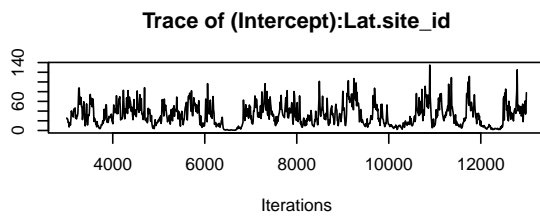
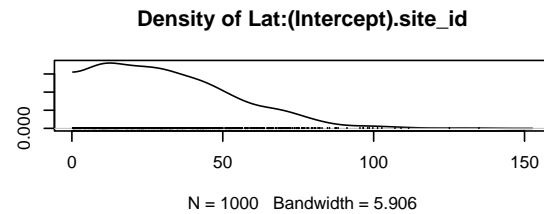
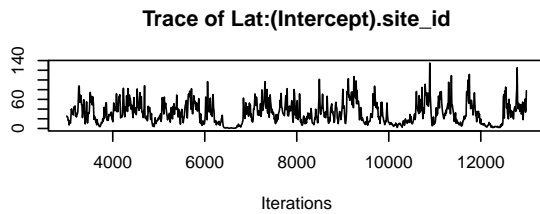
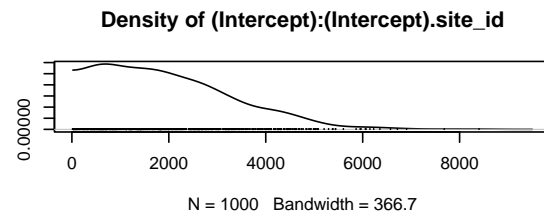
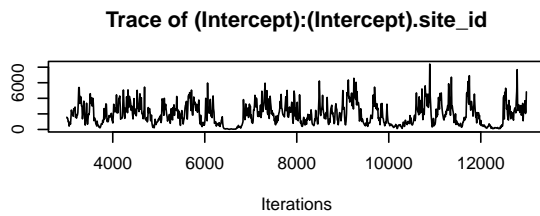
It is important to evaluate the fit of the model. We saw very low effective sample sizes in the model summary, above. We do further model checking below which shows poor mixing of MCMC chains.

```
# The samples from the posterior distribution are stored as mcmc objects,
# which can be summarized and visualized using the coda package

# from MCMC Course notes (page 60):
# Aim to store 1,000-2,000 iterations and have the autocorrelation between
# successive stored iterations less than 0.1 (page 22).

# Assessing model convergence. We do this separately for both fixed
# and random effects. The trace plot should look like a fuzzy caterpillar
# plot(mc1$Sol)

# variances of the random effects (trace plots)
plot(mc1$VCV)
```

*# It looks like some of the variances of the random effects haven't
mixed very well.*

what are the effective sample size for the random effects?

```
coda::effectiveSize(mc1$VCV)
```

```
## (Intercept):(Intercept).site_id      Lat:(Intercept).site_id
##              70.99665                  71.03411
##      (Intercept):Lat.site_id          Lat:Lat.site_id
##              71.03411                  71.11994
##              units
##              1000.00000
```

```
# The effective sample size is very small.
```

```
k = 1 # number of fixed effects
autocorr(mc1$Sol[, 1:k]) # fine - low correlation
```

```
## , , 1
##
##      [,1]
## Lag 0    1.00000000
## Lag 10   -0.02148005
## Lag 50   -0.02817642
## Lag 100  -0.01861508
## Lag 500   0.03884218
```

```
# from MCMC Course notes (page 60):
diag(autocorr(mc1$VCV)[2, , ]) # very high autocorrelation
```

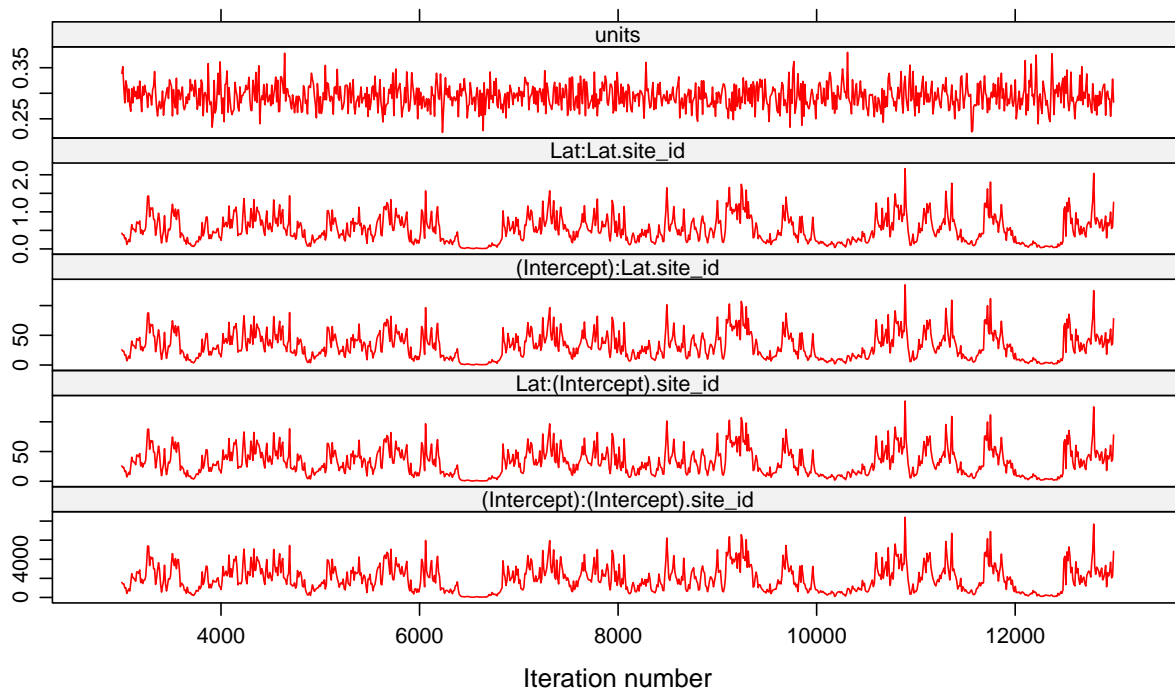
```
## (Intercept):(Intercept).site_id      Lat:(Intercept).site_id
##              0.729329232                  0.726722347
##      (Intercept):Lat.site_id          Lat:Lat.site_id
##              0.726722347                  0.723843392
##              units
##              0.009962174
```

```
# Compare the effective sample sizes between mc2 and mc_Kr
# The rvariance components of the Kruger model has MCMC sampling problems
coda::effectiveSize(mc1$VCV)
```

```
## (Intercept):(Intercept).site_id      Lat:(Intercept).site_id
##              70.99665                  71.03411
##      (Intercept):Lat.site_id          Lat:Lat.site_id
##              71.03411                  71.11994
##              units
##              1000.00000
```

```
# check that the mcmc chain is mixing well - should be "white noise"
#lattice::xyplot(as.mcmc(mc1$Sol), layout=c(6,5), par.strip.text=list(cex=0.5))
```

```
# the variance components
traceK = lattice::xyplot(as.mcmc(mc1$VCV), par.strip.text=list(cex=0.8), col = "red")
traceK
```



1.12 MCMCglmm Random effects (Krüger 2023)

```
sol<-data.frame(mc1$Sol) # random effects
# names(sol)
solm<-reshape2::melt(sol,id.vars=c("X.Intercept.", "season_starting"))
# head(solm)
solm$site_id<-substring(solm$variable,first=22,last=26)
```

The code above drops all the 'Lat.site_id' (these were the slopes) because 'site_id' is blank for them in solm. It keeps only the X.Intercept..site_id. The idea was to plot the slope (decrease in population size), not the intercepts. sigma X.Intercept is the amount of variation in intercepts between sites and sigma Lat would be the amount of variation in the regression slopes between sites. One pointer that helps identify that we are not plotting slopes is that the y-axis is from 0 to 40. Yet, Figure 3B in Krüger 2023 looks similar to the one produced from our own analysis (for the same data). However, later analysis shows that they are not nearly equivalent (the position of the sites are entirely different - see below)

```
ranef = plyr::ddply(solm, c("site_id"), summarise,
  int=mean(value),
  intsd=sd(value),
  intse=intsd/sqrt(length(value)-1))

rlat<-merge(ranef,slopeN,by="site_id")
```

1.13 Predicting counts from mixed model (1960 to 2020) (Krüger 2023)

```
# construct an hypothetical dataframe to generate the populations estimaters
years<-data.frame(season_starting=c(1960:2020))
pops<-data.frame(site_id=countsN$site_id[countsN$ncounts>1],
                 Lat=countsN$Lat[countsN$ncounts>1])
popy<-merge(pops,years)
popy$nested<-c(0) ### MCMCglmm needs a column with the response variable
popypred<-data.frame(predict(mc1,newdata=popy,type="response",
                           marginal=mc1$Random$formula,
                           interval="prediction", posterior="mean"))

popy$fit<-popypred$fit

# WCO: Add lower and upper prediction intervals to the data used for inference
popy$lwr<-popypred$lwr
popy$upr<-popypred$upr
```

The prediction above contains a high degree of uncertainty, which was ignored in the original paper. The uncertainty is the lwr and upr columns, which is the Highest Posterior Density intervals, I believe, from coda::HPDinterval. <https://rdrr.io/cran/MCMCglmm/src/R/predict.MCMCglmm.R>*

Here, the syntax marginal=mc1Randomformula was used. This means random effects were marginalized (see simulation study). posterior="mean" also should not be used, but rather posterior="all".

1.14 How accurate are the predictions relative to observed data?

Let us plot the observed data against predicted data, per site, to see whether observed data and predicted data agree.

```
required_n_pages = round(133/16)+1

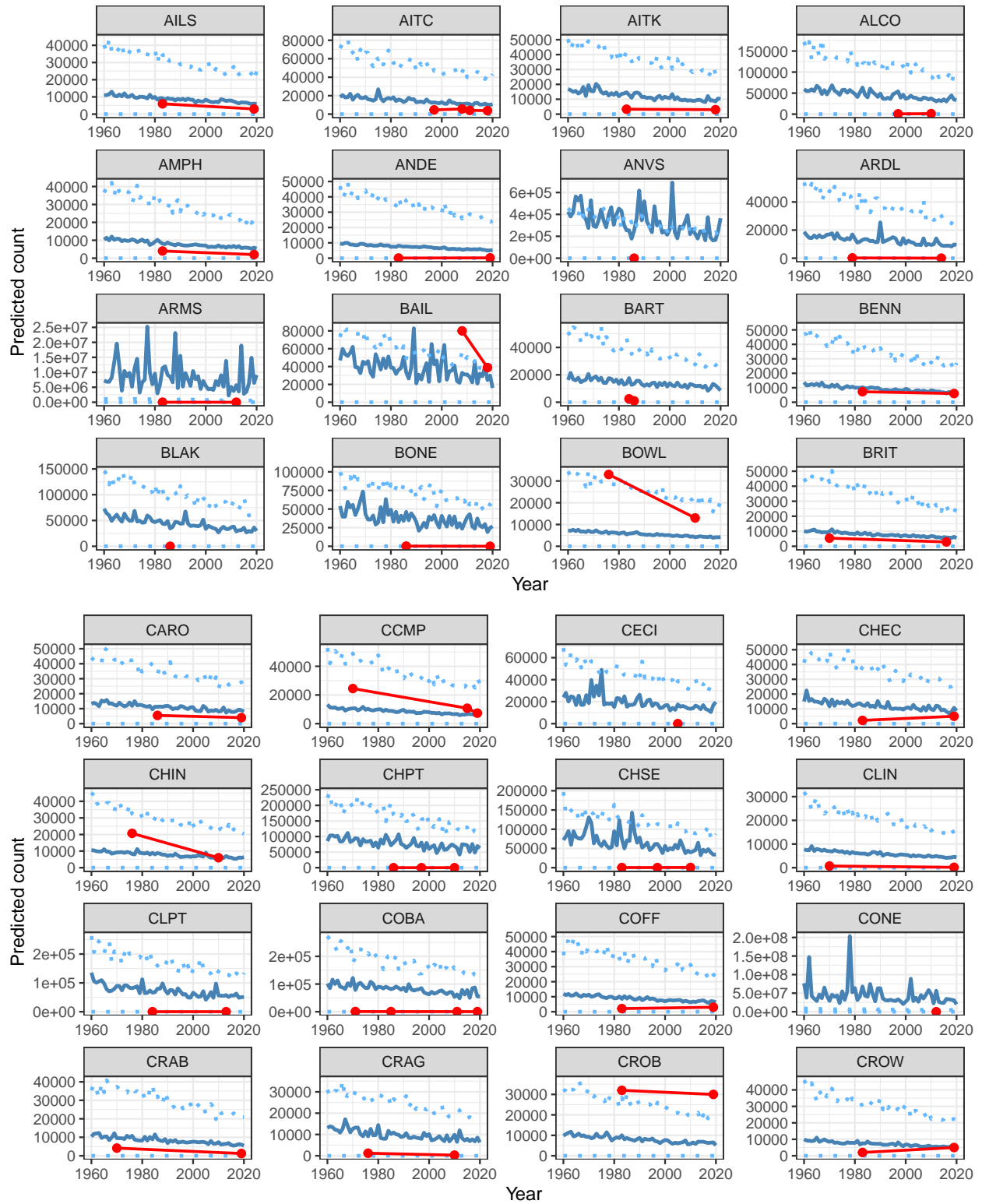
for(i in 1:required_n_pages){

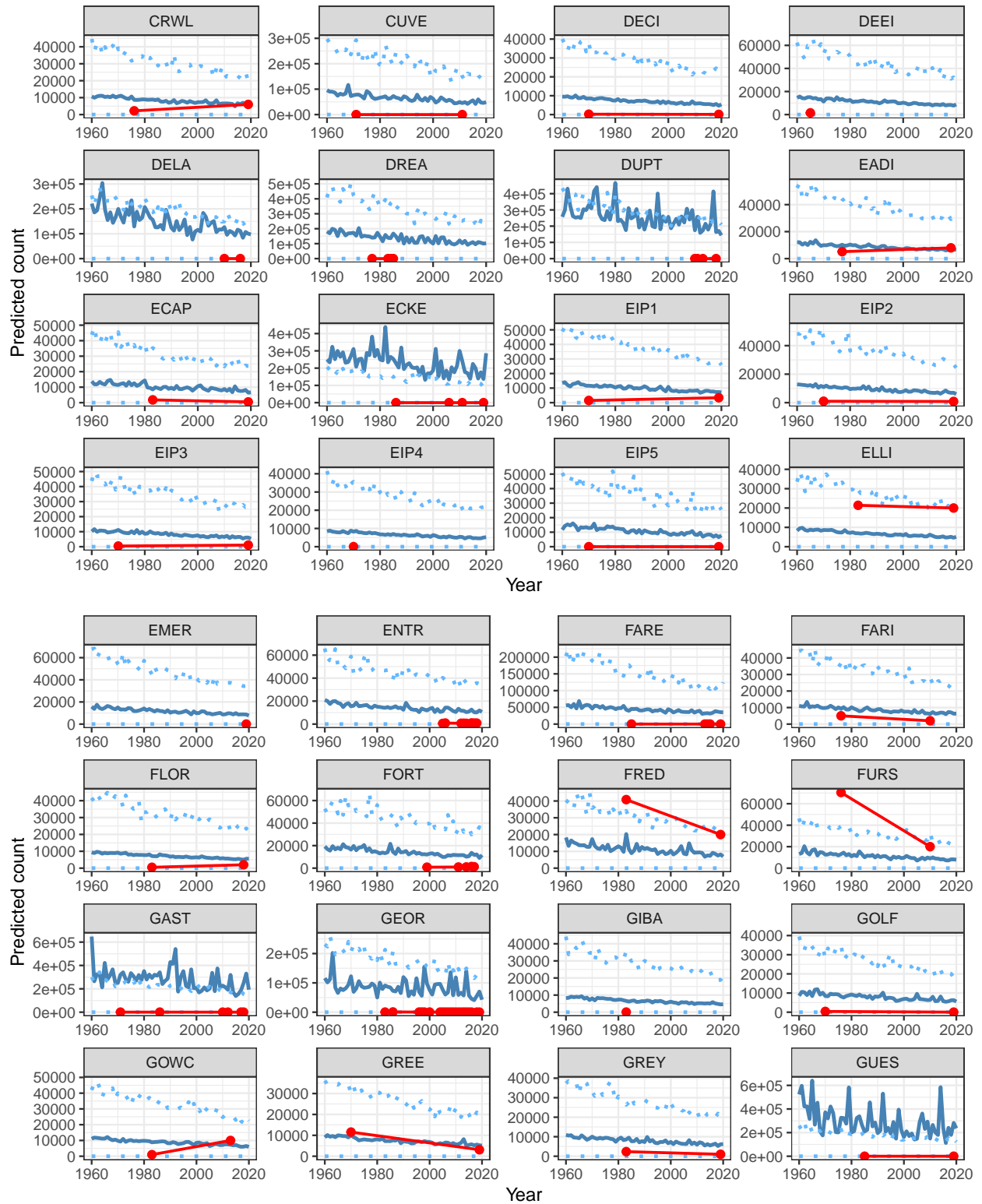
print(ggplot(data = popy) +
  geom_line(aes(x = season_starting, y = fit), col = "steelblue",linewidth=1.04) +
  geom_line(aes(x = season_starting, y = lwr), col = "steelblue1",
            linetype="dotted", linewidth = 1.02) +
  geom_line(aes(x = season_starting, y = upr), col = "steelblue1",
            linetype="dotted",linewidth = 1.02) +
  geom_point(data = nestm3, aes(season_starting, y = nests),
            color = "red", cex = 2) +
  geom_line(data = nestm3, aes(season_starting, y = nests),
            color = "red",linewidth=0.8) +
  theme_bw() +
  xlab("Year") +
  ylab("Predicted count") +
  # theme(strip.text = element_text(size = 1.5)) +
  facet_wrap_paginate(~ site_id, ncol = 4, nrow = 4,
```

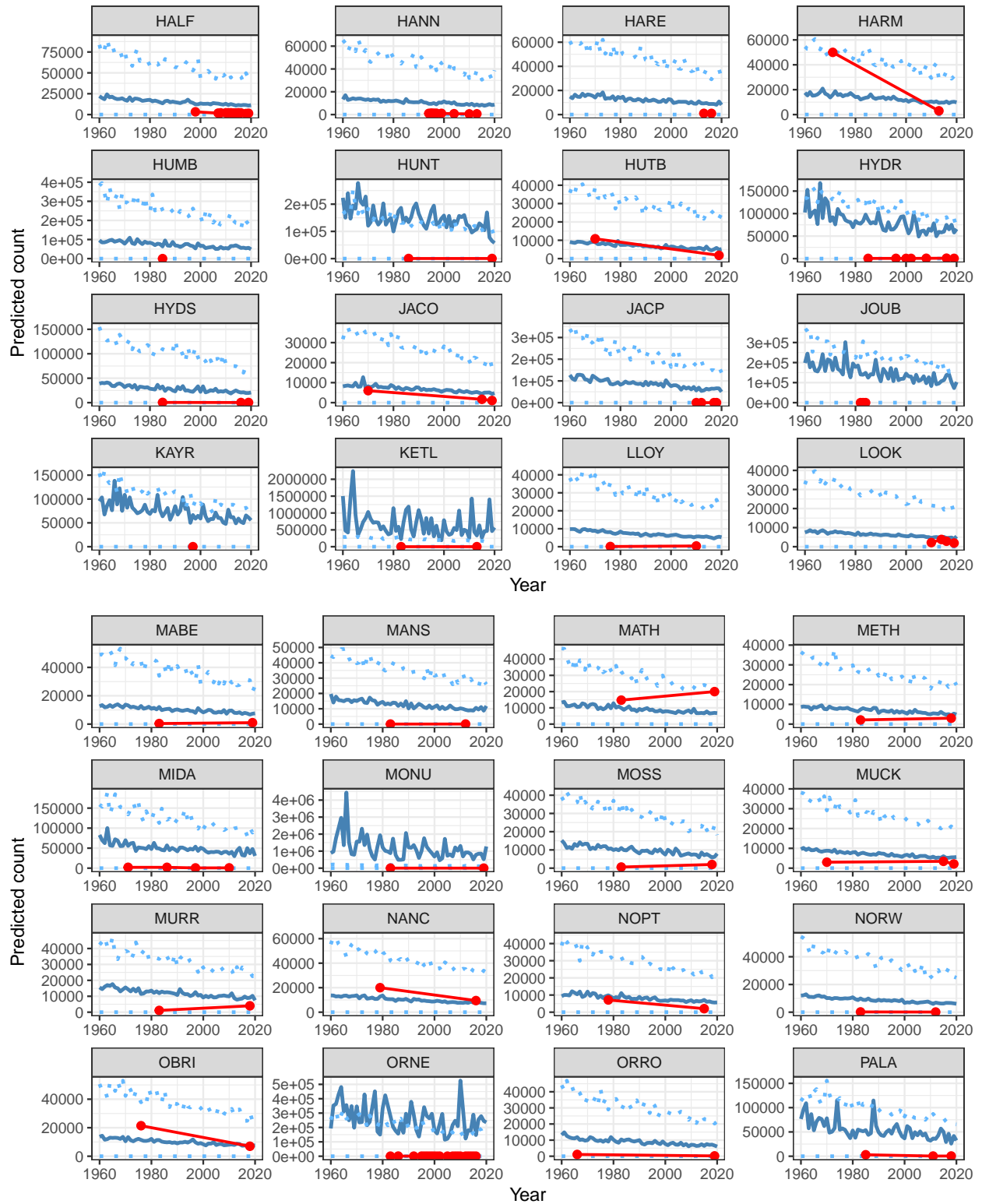
```

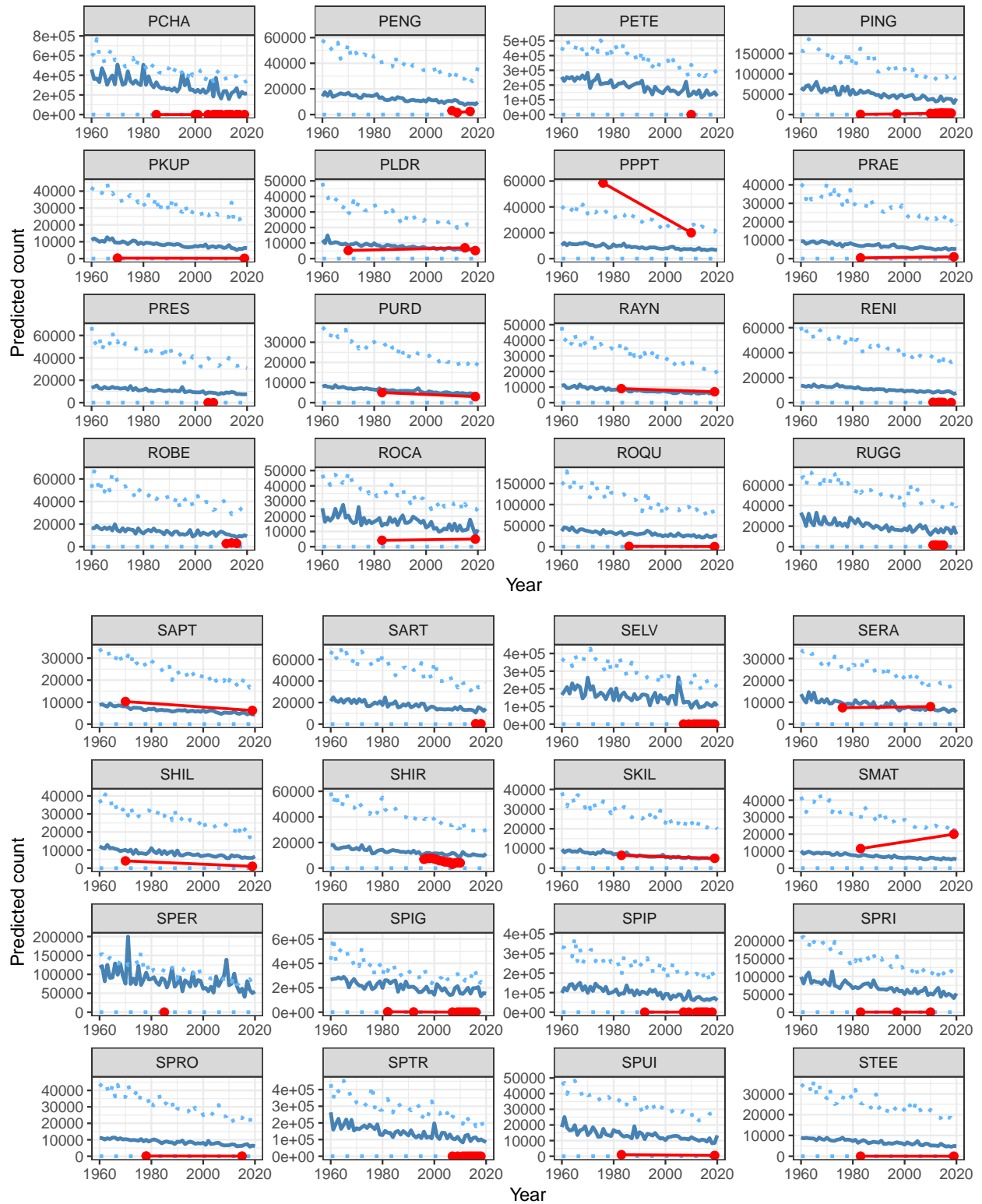
page = i,
scales = 'free'))}

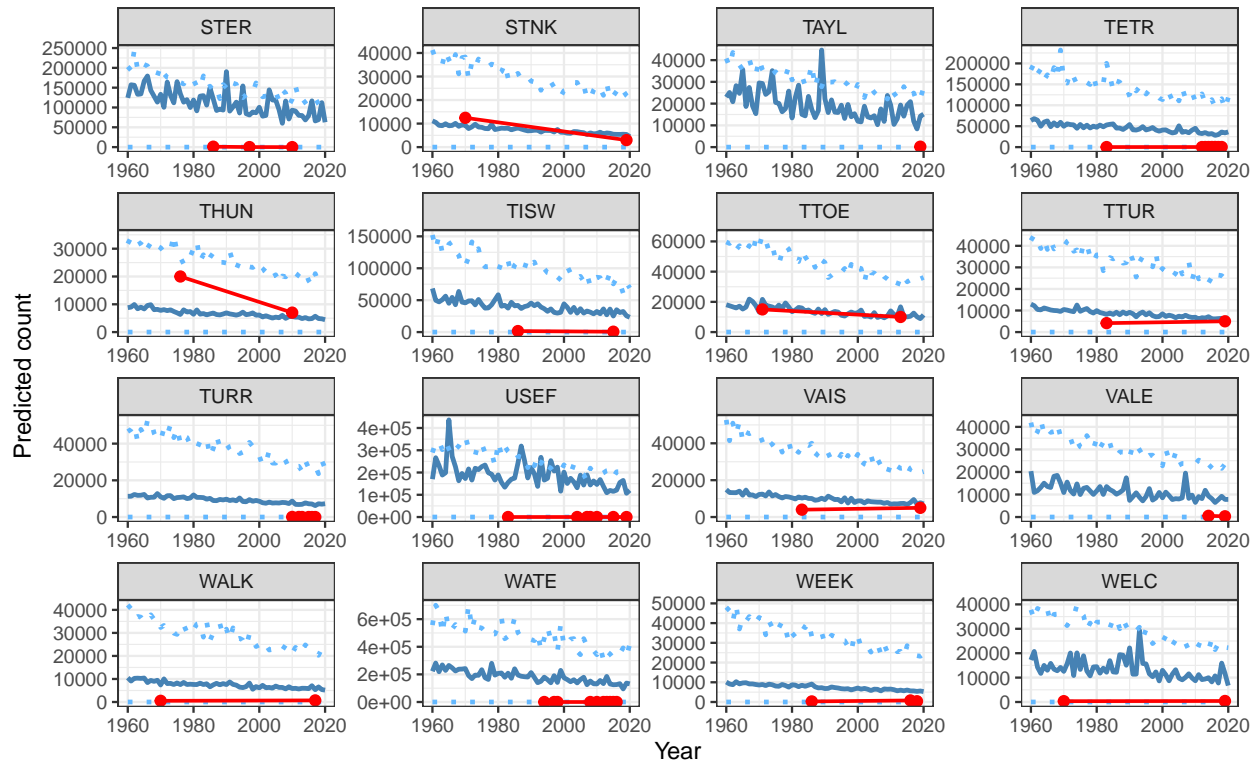
```











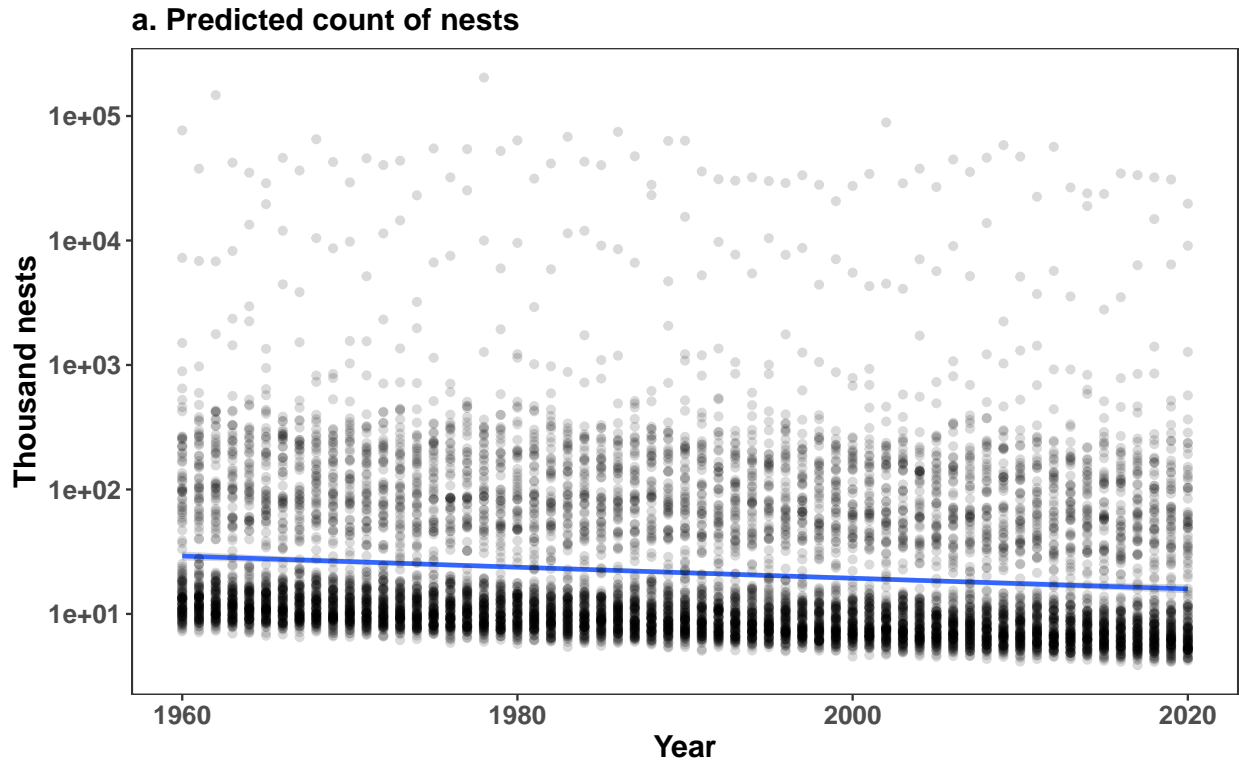
Blue solid lines are the predicted abundance (posterior mean) used by Krüger (2023)
 # to predict regional declines. Light blue dots are the 95 % Highest Posterior Density
 # interval for this prediction. Red points are the observed counts
 # (connected with a red line).

1.15 Figure 3A (Krüger 2023)

```
p1v2<-ggplot(popy,aes(season_starting,fit/1000))+
  geom_smooth()+
  geom_point(alpha=0.15)+xlab("Year")+
  theme_bw()+th+ylib("Thousand nests")+
  ggtitle(label="a. Predicted count of nests")+
  scale_y_log10() # plot from the predicted fit
```

p1v2

```
## 'geom_smooth()' using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



This figure plots all the individual site level predictions. It cannot be sensible given the poor model fit and predictions shown above. On some model runs this figure looks the same as that in Kruger (2023). In other model runs the y-axis scale is much larger (e.g. to $1e+05$). In other words, the model is very unstable (simply re-run and plot the same model multiple times to confirm this fact).

1.16 In the above plots, why are the predicted abundance so ‘wiggly’?

The predictions per site is very ‘wiggly’, which is unexpected for a GLMM model. In addition, the estimated count is sometimes not included in the interval between $\text{popylwr} < -\text{popypredlwr}$ and $\text{popyupr} < -\text{popypredupr}$. Why is this so? Let us review the code used for prediction:

```
popypred<-data.frame(predict(mcl,newdata=popy,type="response", marginal=mclRandomformula,
interval="prediction", posterior="mean"))
```

It is the prediction interval specification that leads to having very ‘wiggly’ predictions. Prediction intervals are obtained through posterior predictive simulation. Note, however, that in the reanalysis (below) we also specify prediction intervals (to agree with this code) but in that case the predictions do not have such extreme wiggleness, and well-behaved credible intervals.

Let us plot “confidence” (credible intervals) rather than prediction intervals <https://www.rdocumentation.org/packages/MCMCglmm/versions/2.35/topics/predict.MCMCglmm>

2 Change prediction to ‘confidence’ (credible) interval

```
# you have to change posterior="mean" to posterior="mean" when using interval="confidence", otherwise
# there is an error about HPDinterval.mcmc(mcmc(post.pred), prob = level) : obj must have nsamp > 1
popypred <-data.frame(predict(mc1,newdata=popy,type="response",
                             marginal=mc1$Random$formula,
                             interval="confidence", posterior="all"))

popy$fit<-popypred$fit
# Add lower and upper prediction intervals to the data used for inference
popy$lwr<-popypred$lwr
popy$upr<-popypred$upr

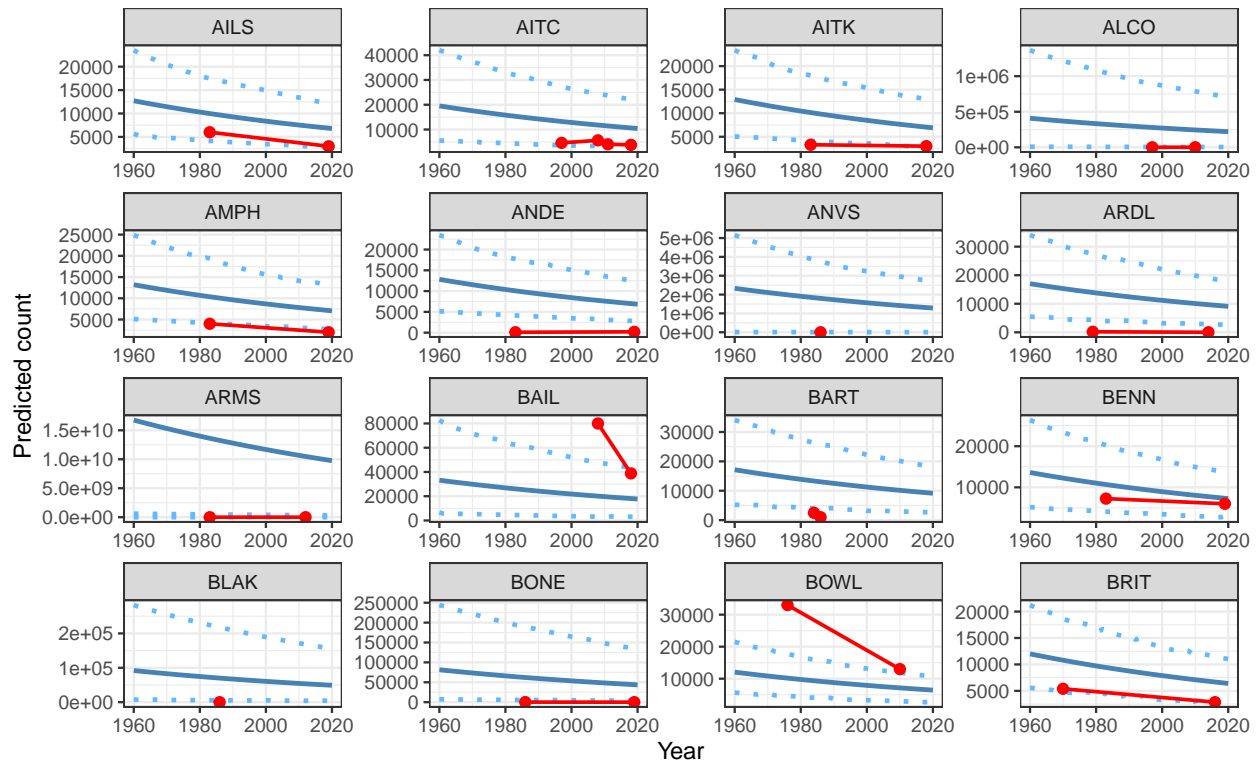
required_n_pages = round(133/16)+1

for(i in 1:required_n_pages){

print(ggplot(data = popy) +
      geom_line(aes(x = season_starting, y = fit), col = "steelblue",linewidth=1.04) +
      geom_line(aes(x = season_starting, y = lwr), col = "steelblue1",
                linetype="dotted", linewidth = 1.02) +
      geom_line(aes(x = season_starting, y = upr), col = "steelblue1",
                linetype="dotted",linewidth = 1.02) +
      geom_point(data = nestm3, aes(season_starting, y = nests),
                 color = "red", cex = 2) +
      geom_line(data = nestm3, aes(season_starting, y = nests),
                 color = "red",linewidth=0.8) +
      theme_bw() +
      xlab("Year") +
      ylab("Predicted count") +
      # theme(strip.text = element_text(size = 1.5)) +
      facet_wrap_paginate(~ site_id, ncol = 4, nrow = 4,
                          page = i,
                          scales = 'free'))}
```

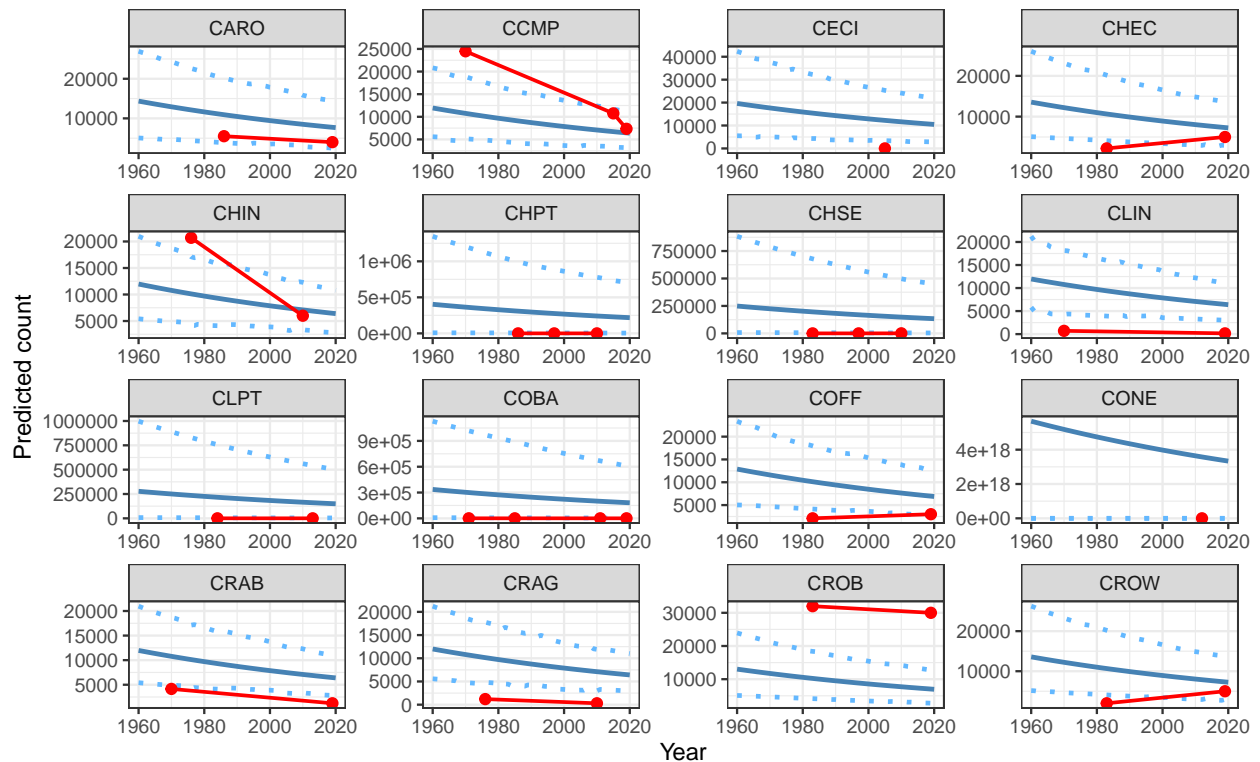
```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

```
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



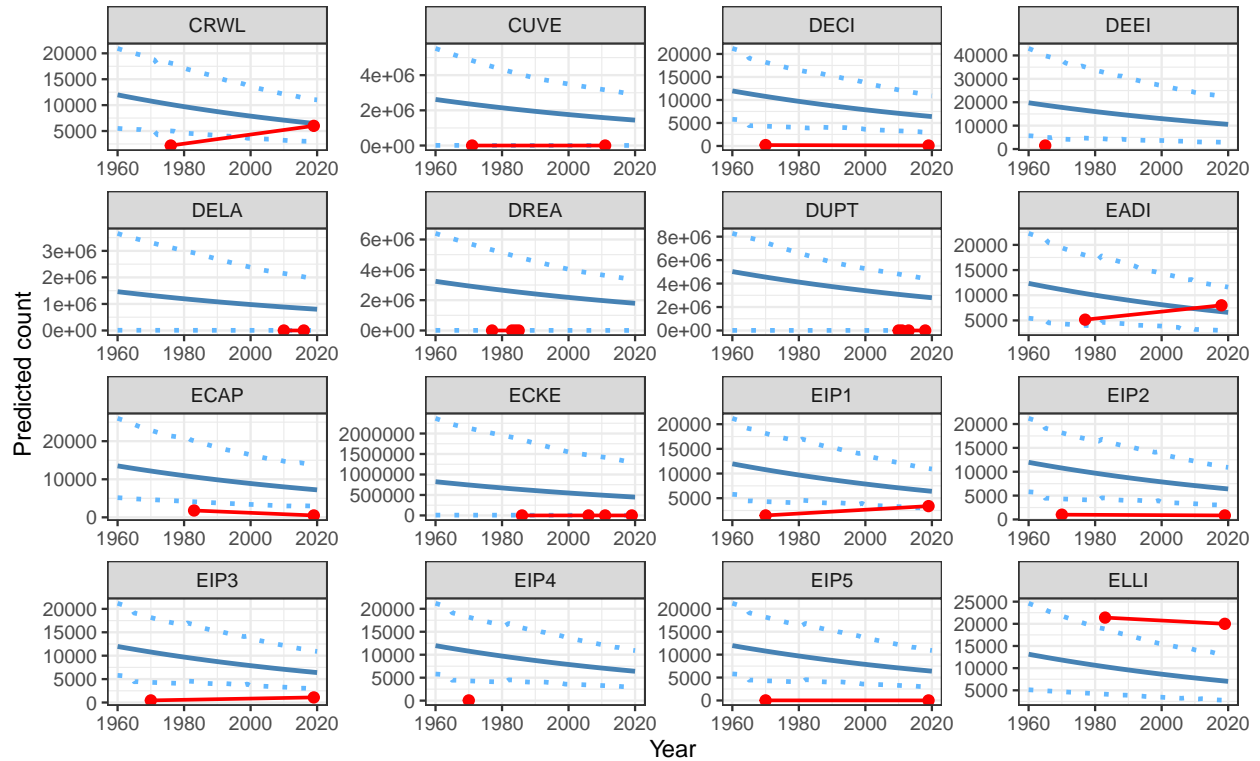
```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



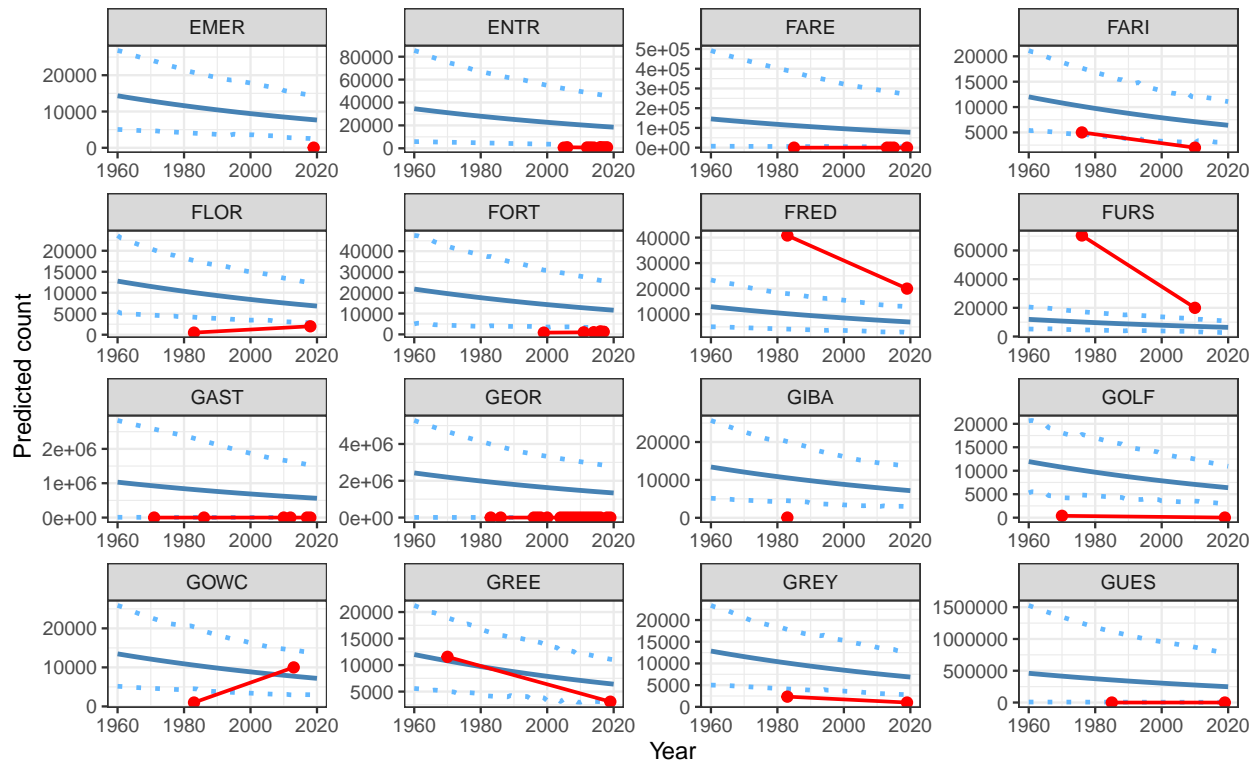
```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

```
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

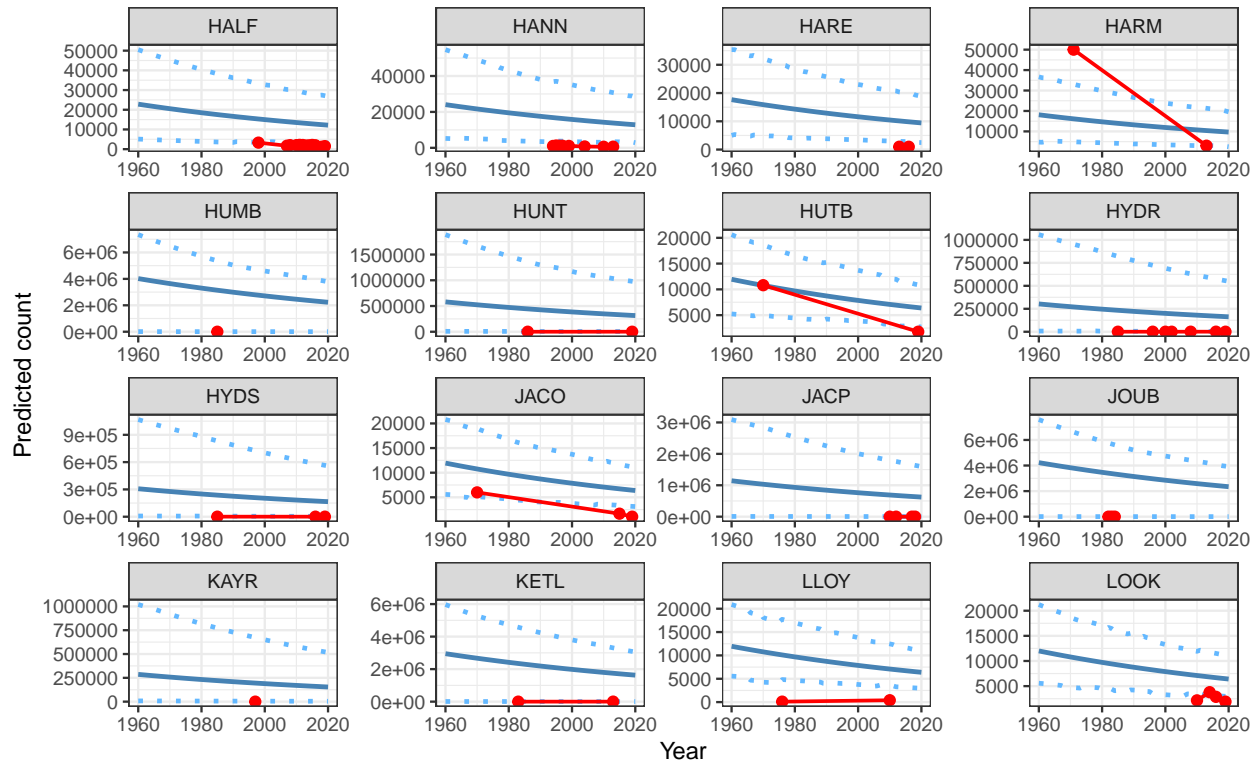
```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



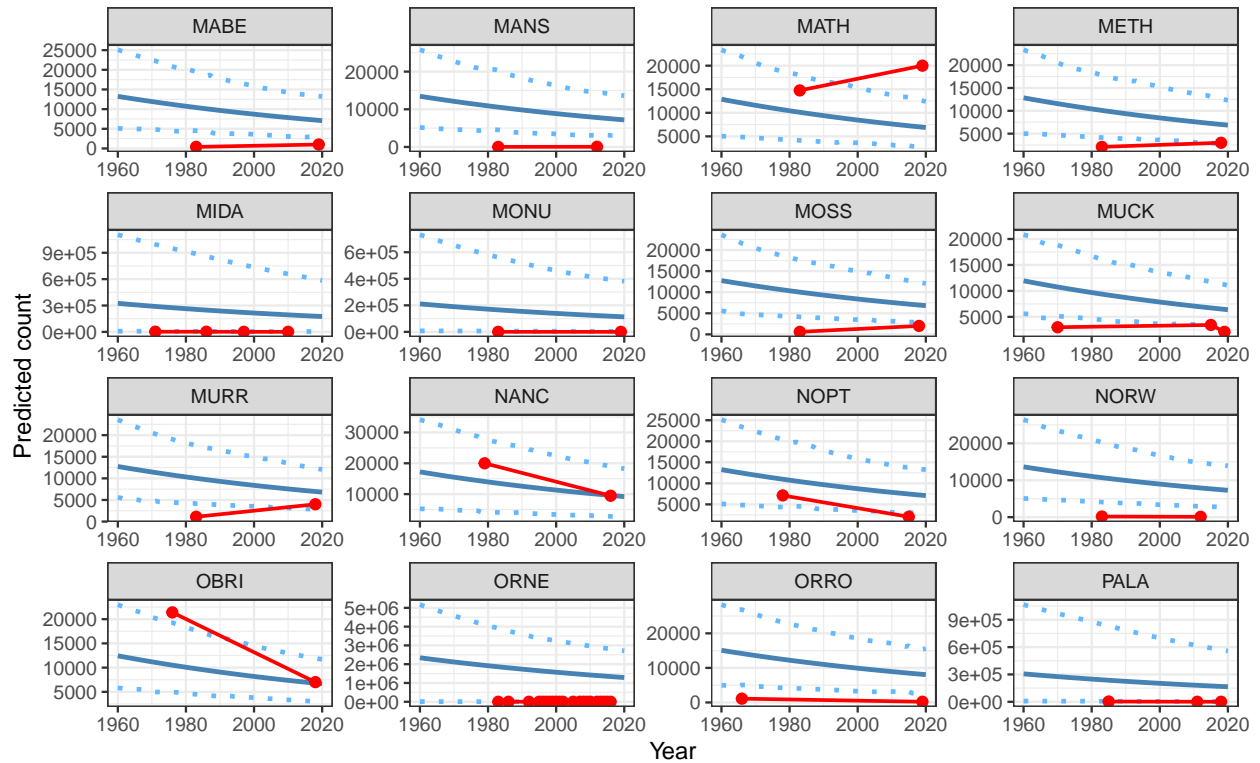
```
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

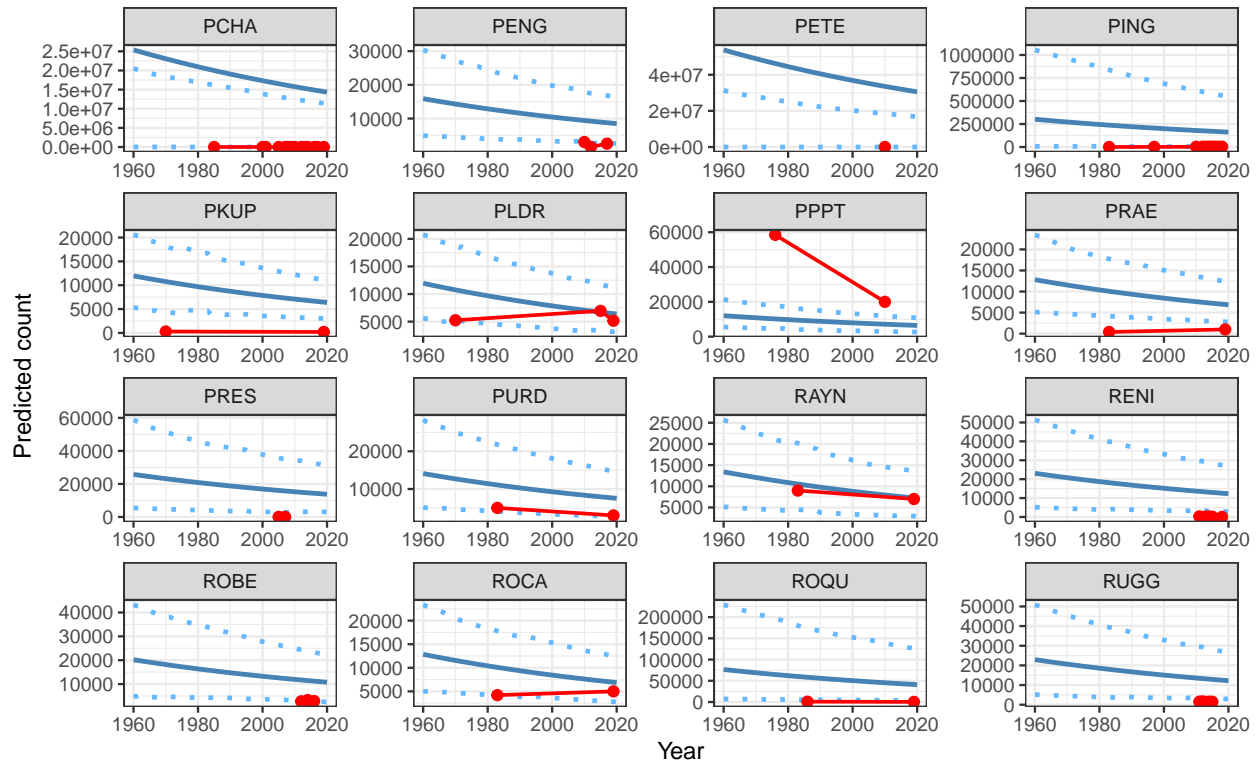


```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



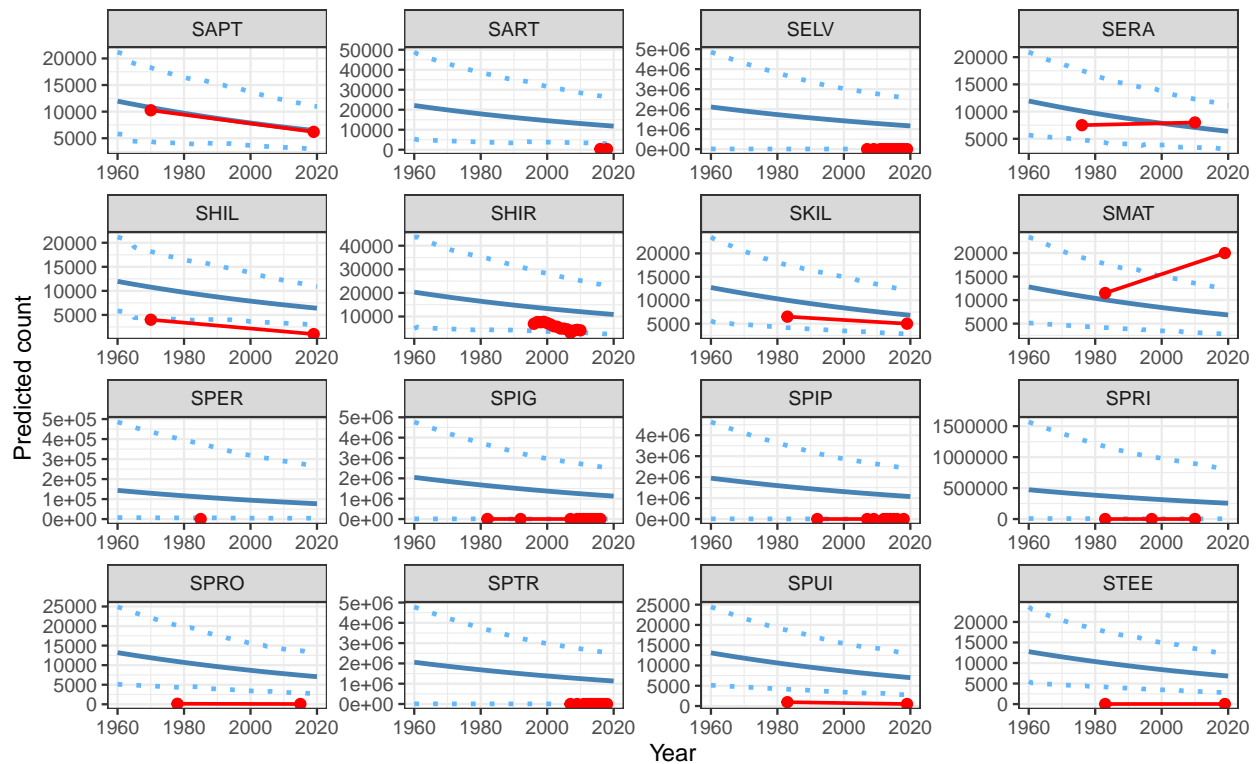
```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

```
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



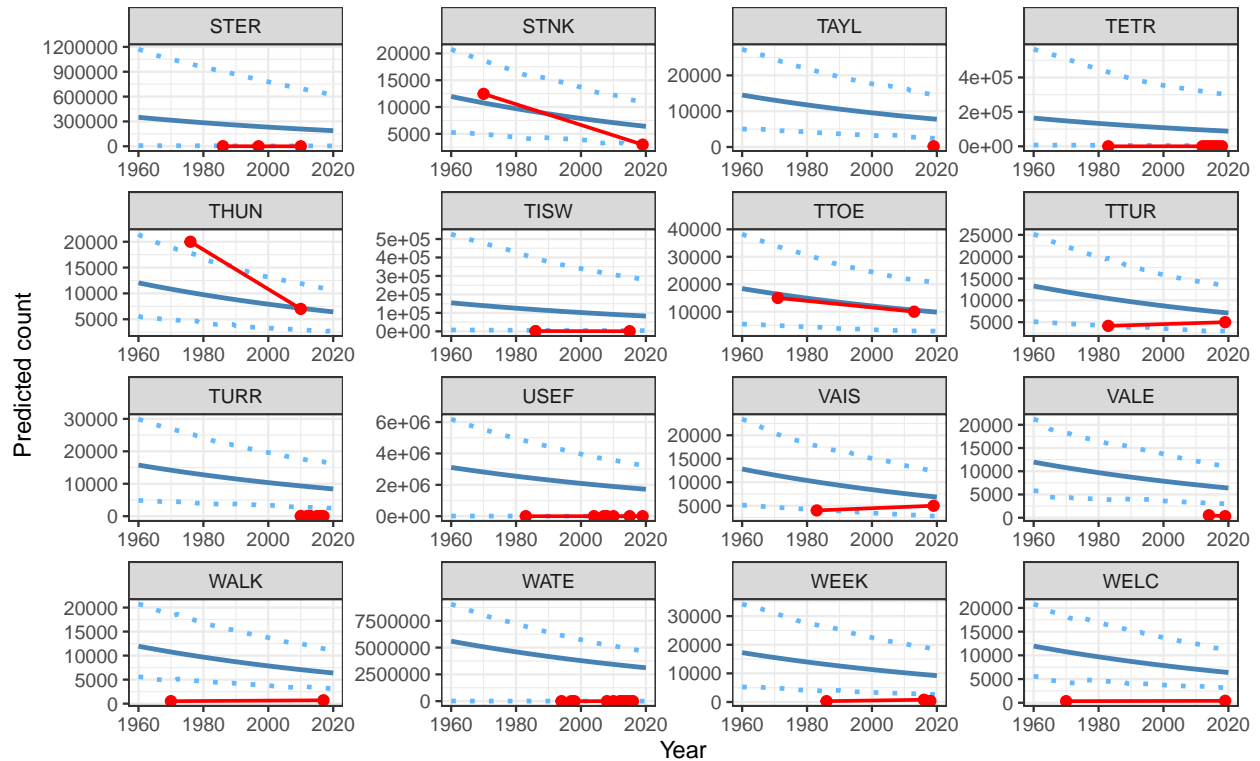
```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



```
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
## 'geom_line()': Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

i Do you need to adjust the group aesthetic?



Blue solid lines are the predicted abundance (posterior mean) used by Krüger (2023)
 # to predict regional declines. Light blue dots are the 95 % Highest Posterior Density
 # interval for this prediction. Red points are the observed counts
 # (connected with a red line).

Extreme / strange values for prediction intervals are perhaps expected given the large
 # l-95% CI u-95% CI in the model:

`summary(mcl)`

```
##
## Iterations = 3001:12991
## Thinning interval = 10
## Sample size = 1000
##
## DIC: 4785.787
##
## G-structure: ~us(1 + Lat):site_id
##
##               post.mean 1-95% CI u-95% CI eff.samp
## (Intercept):(Intercept).site_id 1991.5479 28.490585 4546.266    71.00
## Lat:(Intercept).site_id         32.4988  0.255875   73.122    71.03
## (Intercept):Lat.site_id         32.4988  0.255875   73.122    71.03
## Lat:Lat.site_id                 0.5313  0.006634    1.185    71.12
##
## R-structure: ~units
```

```
##
##      post.mean l-95% CI u-95% CI eff.samp
## units    0.2941  0.2539  0.3441    1000
##
## Location effects: nests ~ season_starting
##
##      post.mean  l-95% CI  u-95% CI  eff.samp  pMCMC
## (Intercept)    27.743045 20.730480 34.391312    824.5 <0.001 ***
## season_starting -0.010443 -0.013982 -0.007111    823.3 <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.1 Figure 3B (Krüger 2023): Latitude

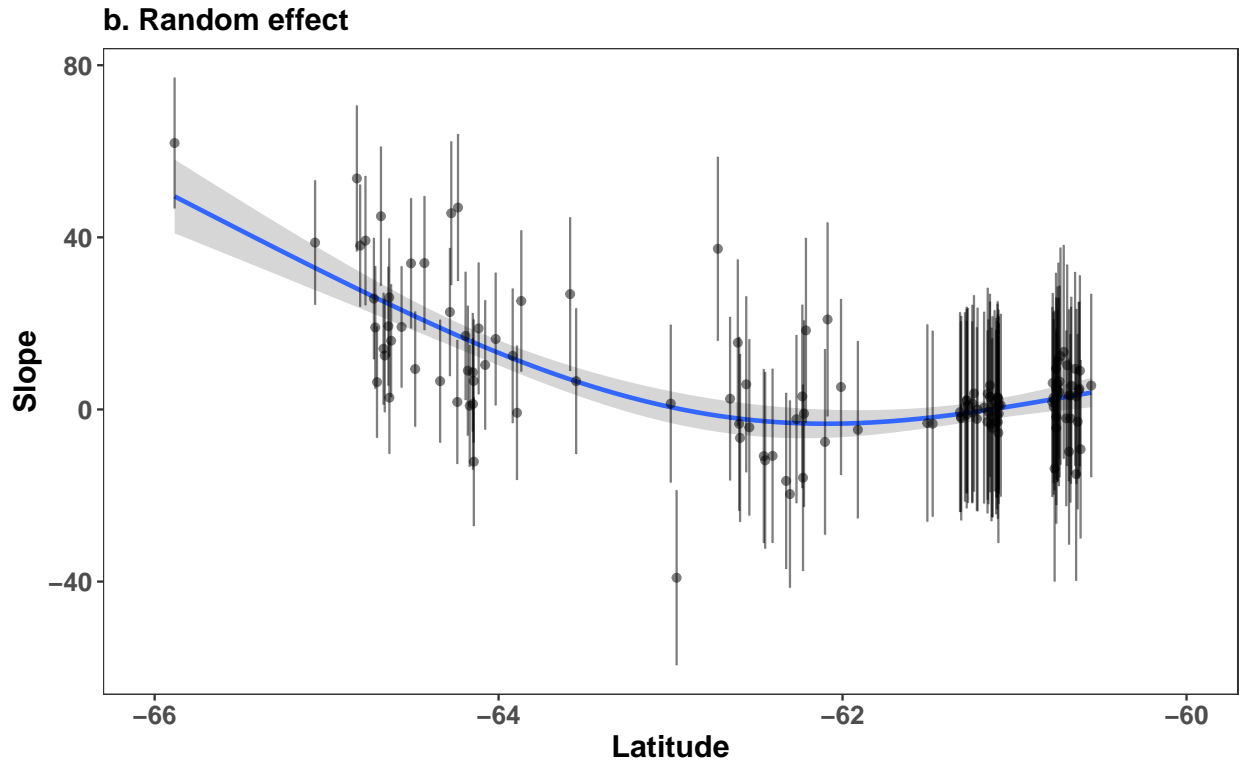
This plots the MCMCglmm intercept - it is (even) labelled “int” here. But the paper legend says slope (which is what we are interested in). This figure makes use of a very poor fitting model (mc1), but initially the output looks similar to that from our own analysis. That is because both plots latitude on the x-axis - so the distribution of points on the x-axis are the same. The sites vary a lot on the y-axis. This plot does not represent changes in population rate of change.

The error bar is calculated as $sd/2$. The paper caption refers to ‘standard deviation’ But why divide the standard deviation by 2?

```
p2<- ggplot(subset(rlat,Lat>(-67)),aes(Lat,int))+
  stat_smooth(method="gam",formula=y~s(x,k=2))+
  geom_errorbar(aes(ymin=int-(intsd/2),ymax=int+(intsd/2)),alpha=0.5)+
  geom_point(alpha=0.5)+
  theme_bw()+th+
  ggtitle(label="b. Random effect")+
  ylab("Slope")+xlim(-66,-60)+
  xlab("Latitude")
```

p2

```
## Warning in smooth.construct.tp.smooth.spec(object, dk$data, dk$knots): basis dimension, k, increased
```



*# ps: as results are based on randomization
 # expect slight differences every time you run the model
 # but the trends are consistent everytime
 # lagged analysis to determine how much pops have decreased*

2.2 Population change in 3 generations

We did not consider this part of the Krüger (2023) analysis, as it is dependent on the above predictions to be reasonable approximations of (observed) abundance. Furthermore, these calculations do not account for any uncertainty in the predicted abundance, which is not a good protocol to follow.

```
library(lubridate)
library(tidyr)
#library(tidyquant)
library(dplyr)
library(broom)
library(purrr)
library(stringr)
library(knitr)
#library(timetk)
```

Use library(xts) instead, below:

```
head(poppy)
```

```
##   site_id   Lat season_starting nests   fit   lwr   upr
```

```
## 1    AILS -60.780      1960    0 12716.07 5650.993 23519.07
## 2    AITC -62.407      1960    0 19529.33 5534.547 42028.08
## 3    AITK -60.738      1960    0 12908.88 5038.378 23387.33
## 4    ALCO -64.240      1960    0 408943.37 7906.923 1365344.19
## 5    AMPH -60.684      1960    0 13196.95 5117.787 24876.05
## 6    ANDE -60.757      1960    0 12818.42 5145.752 23461.71
```

```
popT<-ddply(popy, c("season_starting"), summarise,
            tot=sum(fit), ### total population
            mean=mean(fit)) ### mean population

# create a time stamp for year
popT$TS<-(as.POSIXct(strptime(paste(popT$season_starting,c("01-01"),sep="-"),
                                format="%Y-%m-%d",tz="GMT"))) )

# create a time stamp for year
popy$TS<-(as.POSIXct(strptime(paste(popy$season_starting,c("01-01"),sep="-"),
                                format="%Y-%m-%d",tz="GMT"))) )

mts<-xts::xts(popT$tot,order.by=popT$TS) # create a temporal data frame

# create a lag data frame
mlag<-((data.frame(year=popT$season_starting,mts %>%
                    xts::lag.xts(k = c(0,27,28,29,30)))))
mlag
```

```
##      year      lag0      lag27      lag28      lag29
## 1960-01-01 1960 5.659570e+18      NA      NA      NA
## 1961-01-01 1961 5.609868e+18      NA      NA      NA
## 1962-01-01 1962 5.560603e+18      NA      NA      NA
## 1963-01-01 1963 5.511771e+18      NA      NA      NA
## 1964-01-01 1964 5.463368e+18      NA      NA      NA
## 1965-01-01 1965 5.415390e+18      NA      NA      NA
## 1966-01-01 1966 5.367833e+18      NA      NA      NA
## 1967-01-01 1967 5.320695e+18      NA      NA      NA
## 1968-01-01 1968 5.273971e+18      NA      NA      NA
## 1969-01-01 1969 5.227657e+18      NA      NA      NA
## 1970-01-01 1970 5.181750e+18      NA      NA      NA
## 1971-01-01 1971 5.136246e+18      NA      NA      NA
## 1972-01-01 1972 5.091143e+18      NA      NA      NA
## 1973-01-01 1973 5.046435e+18      NA      NA      NA
## 1974-01-01 1974 5.002120e+18      NA      NA      NA
## 1975-01-01 1975 4.958195e+18      NA      NA      NA
## 1976-01-01 1976 4.914656e+18      NA      NA      NA
## 1977-01-01 1977 4.871499e+18      NA      NA      NA
## 1978-01-01 1978 4.828721e+18      NA      NA      NA
## 1979-01-01 1979 4.786319e+18      NA      NA      NA
## 1980-01-01 1980 4.744290e+18      NA      NA      NA
## 1981-01-01 1981 4.702630e+18      NA      NA      NA
## 1982-01-01 1982 4.661336e+18      NA      NA      NA
## 1983-01-01 1983 4.620404e+18      NA      NA      NA
## 1984-01-01 1984 4.579833e+18      NA      NA      NA
## 1985-01-01 1985 4.539617e+18      NA      NA      NA
## 1986-01-01 1986 4.499755e+18      NA      NA      NA
## 1987-01-01 1987 4.460244e+18 5.659570e+18      NA      NA
```

```

## 1988-01-01 1988 4.421079e+18 5.609868e+18 5.659570e+18 NA
## 1989-01-01 1989 4.382258e+18 5.560603e+18 5.609868e+18 5.659570e+18
## 1990-01-01 1990 4.343779e+18 5.511771e+18 5.560603e+18 5.609868e+18
## 1991-01-01 1991 4.305637e+18 5.463368e+18 5.511771e+18 5.560603e+18
## 1992-01-01 1992 4.267831e+18 5.415390e+18 5.463368e+18 5.511771e+18
## 1993-01-01 1993 4.230357e+18 5.367833e+18 5.415390e+18 5.463368e+18
## 1994-01-01 1994 4.193212e+18 5.320695e+18 5.367833e+18 5.415390e+18
## 1995-01-01 1995 4.156393e+18 5.273971e+18 5.320695e+18 5.367833e+18
## 1996-01-01 1996 4.119897e+18 5.227657e+18 5.273971e+18 5.320695e+18
## 1997-01-01 1997 4.083723e+18 5.181750e+18 5.227657e+18 5.273971e+18
## 1998-01-01 1998 4.047866e+18 5.136246e+18 5.181750e+18 5.227657e+18
## 1999-01-01 1999 4.012324e+18 5.091143e+18 5.136246e+18 5.181750e+18
## 2000-01-01 2000 3.977094e+18 5.046435e+18 5.091143e+18 5.136246e+18
## 2001-01-01 2001 3.942174e+18 5.002120e+18 5.046435e+18 5.091143e+18
## 2002-01-01 2002 3.907560e+18 4.958195e+18 5.002120e+18 5.046435e+18
## 2003-01-01 2003 3.873251e+18 4.914656e+18 4.958195e+18 5.002120e+18
## 2004-01-01 2004 3.839243e+18 4.871499e+18 4.914656e+18 4.958195e+18
## 2005-01-01 2005 3.805533e+18 4.828721e+18 4.871499e+18 4.914656e+18
## 2006-01-01 2006 3.772120e+18 4.786319e+18 4.828721e+18 4.871499e+18
## 2007-01-01 2007 3.739000e+18 4.744290e+18 4.786319e+18 4.828721e+18
## 2008-01-01 2008 3.706171e+18 4.702630e+18 4.744290e+18 4.786319e+18
## 2009-01-01 2009 3.673631e+18 4.661336e+18 4.702630e+18 4.744290e+18
## 2010-01-01 2010 3.641376e+18 4.620404e+18 4.661336e+18 4.702630e+18
## 2011-01-01 2011 3.609405e+18 4.579833e+18 4.620404e+18 4.661336e+18
## 2012-01-01 2012 3.577715e+18 4.539617e+18 4.579833e+18 4.620404e+18
## 2013-01-01 2013 3.546303e+18 4.499755e+18 4.539617e+18 4.579833e+18
## 2014-01-01 2014 3.515166e+18 4.460244e+18 4.499755e+18 4.539617e+18
## 2015-01-01 2015 3.484304e+18 4.421079e+18 4.460244e+18 4.499755e+18
## 2016-01-01 2016 3.453712e+18 4.382258e+18 4.421079e+18 4.460244e+18
## 2017-01-01 2017 3.423389e+18 4.343779e+18 4.382258e+18 4.421079e+18
## 2018-01-01 2018 3.393333e+18 4.305637e+18 4.343779e+18 4.382258e+18
## 2019-01-01 2019 3.363540e+18 4.267831e+18 4.305637e+18 4.343779e+18
## 2020-01-01 2020 3.334009e+18 4.230357e+18 4.267831e+18 4.305637e+18
##
## lag30
## 1960-01-01 NA
## 1961-01-01 NA
## 1962-01-01 NA
## 1963-01-01 NA
## 1964-01-01 NA
## 1965-01-01 NA
## 1966-01-01 NA
## 1967-01-01 NA
## 1968-01-01 NA
## 1969-01-01 NA
## 1970-01-01 NA
## 1971-01-01 NA
## 1972-01-01 NA
## 1973-01-01 NA
## 1974-01-01 NA
## 1975-01-01 NA
## 1976-01-01 NA
## 1977-01-01 NA
## 1978-01-01 NA
## 1979-01-01 NA

```



```
## 1980-01-01      NA
## 1981-01-01      NA
## 1982-01-01      NA
## 1983-01-01      NA
## 1984-01-01      NA
## 1985-01-01      NA
## 1986-01-01      NA
## 1987-01-01      NA
## 1988-01-01      NA
## 1989-01-01      NA
## 1990-01-01 5.659570e+18
## 1991-01-01 5.609868e+18
## 1992-01-01 5.560603e+18
## 1993-01-01 5.511771e+18
## 1994-01-01 5.463368e+18
## 1995-01-01 5.415390e+18
## 1996-01-01 5.367833e+18
## 1997-01-01 5.320695e+18
## 1998-01-01 5.273971e+18
## 1999-01-01 5.227657e+18
## 2000-01-01 5.181750e+18
## 2001-01-01 5.136246e+18
## 2002-01-01 5.091143e+18
## 2003-01-01 5.046435e+18
## 2004-01-01 5.002120e+18
## 2005-01-01 4.958195e+18
## 2006-01-01 4.914656e+18
## 2007-01-01 4.871499e+18
## 2008-01-01 4.828721e+18
## 2009-01-01 4.786319e+18
## 2010-01-01 4.744290e+18
## 2011-01-01 4.702630e+18
## 2012-01-01 4.661336e+18
## 2013-01-01 4.620404e+18
## 2014-01-01 4.579833e+18
## 2015-01-01 4.539617e+18
## 2016-01-01 4.499755e+18
## 2017-01-01 4.460244e+18
## 2018-01-01 4.421079e+18
## 2019-01-01 4.382258e+18
## 2020-01-01 4.343779e+18
```

```
# proportional change for all lags
mlag$ch3<-(mlag$lag0/mlag$lag27)-1
mlag$ch4<-(mlag$lag0/mlag$lag28)-1
mlag$ch5<-(mlag$lag0/mlag$lag29)-1
mlag$ch6<-(mlag$lag0/mlag$lag30)-1

mlags<-data.frame(year=mlag$year,mlag[7:10])

chm<-na.omit(melt(mlags,id.vars="year"))
summary(chm$value)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
## -0.2325 -0.2257 -0.2188 -0.2220 -0.2119 -0.2119
```

```
quantile(chm$value,probs=0.95)
```

```
##          95%  
## -0.2118895
```

```
quantile(chm$value,probs=0.05)
```

```
##          5%  
## -0.2324837
```

```
mean(chm$value)
```

```
## [1] -0.2219533
```

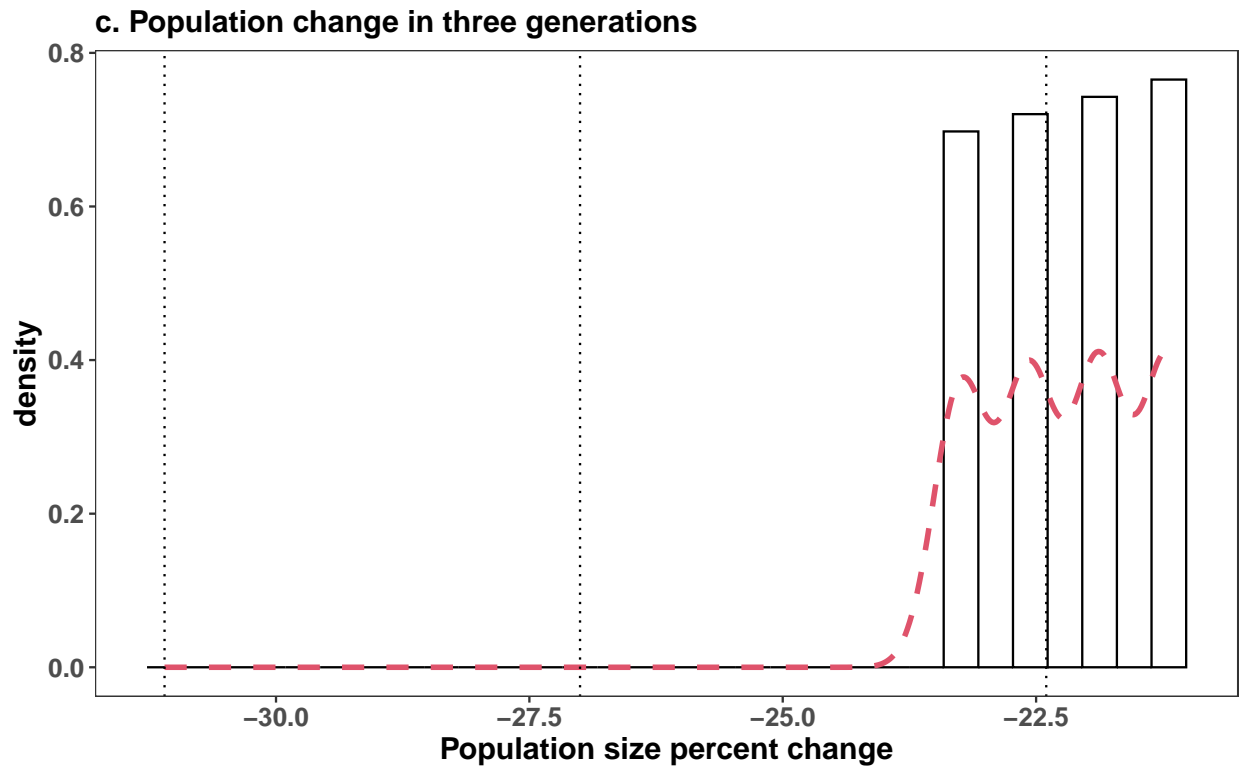
```
sd(chm$value)
```

```
## [1] 0.007695194
```

```
p3<-ggplot(chm,aes(value*100))+  
  geom_histogram(aes(y = ..density..), colour = 1, fill = "white") +  
  geom_density(lwd = 1.2, linetype = 2,colour = 2)+  
  theme_bw()+th+  
  geom_vline(xintercept = c(-22.4,-27.0,-31.1),linetype="dotted")+  
  xlab("Population size percent change")+  
  ggtitle(label="c. Population change in three generations")
```

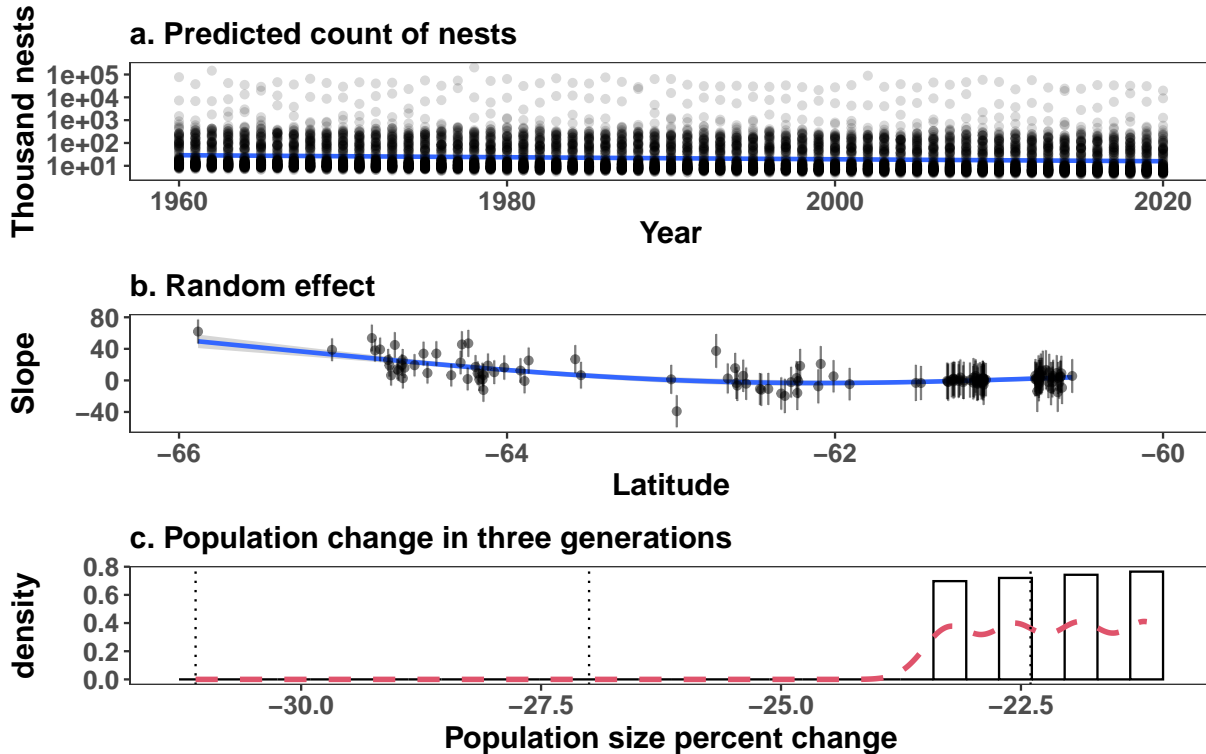
```
p3
```

```
## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```



```
p1v2/p2/p3
```

```
## 'geom_smooth()' using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'  
## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```



3 Reanalysis for Oosthuizen et al.

3.1 Fit a better GLMM

How is this model different to Kruger (2023)? Here, we used the same data, but we: 1) used a different model specification for fixed and random effects 2) z-standardized the covariates before running the model 3) used longer mcmc chains 4) when predicting from the fitted model, we did not marginalise the random effects

```
# head(nestM3)
```

```
# Covariates should be standardized.
```

```
nestM3$Zseason_starting = scale(nestM3$season_starting)
nestM3$ZLat = scale(nestM3$Lat)
```

```
mc2 <- MCMCglmm(nests ~ Zseason_starting * ZLat,
  random=~us(1 + Zseason_starting):site_id,
  rcov=~units,
  family="poisson", mev=NULL,
  data=nestM3, start=NULL, nodes="ALL", scale=TRUE,
  nitt=23000, thin=10, burnin=13000, pr=T,
  pl=FALSE, verbose=TRUE, DIC=TRUE, singular.ok=FALSE, saveX=TRUE,
  prior=prior, saveZ=TRUE, saveXL=TRUE, slice=FALSE,
  ginverse=NULL, trunc=FALSE)
```

```
summary(mc2)
```

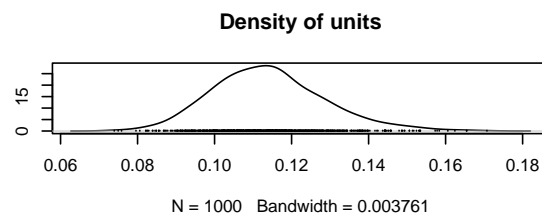
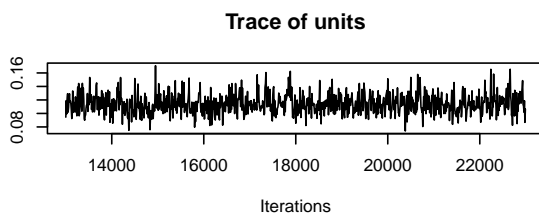
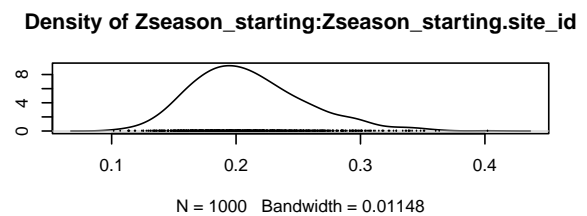
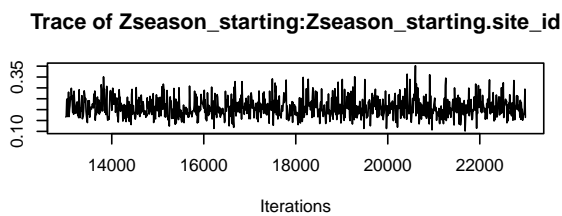
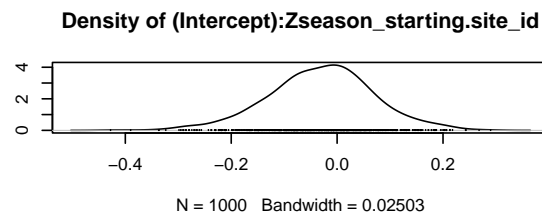
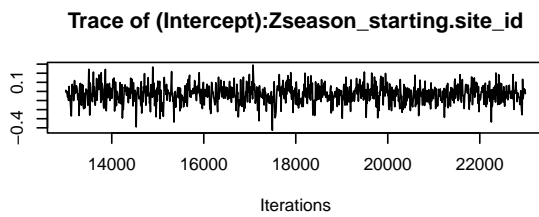
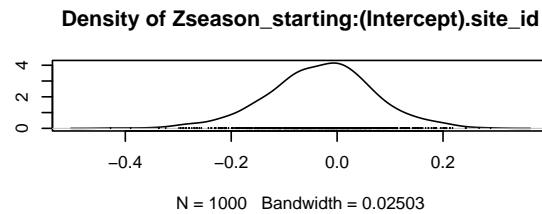
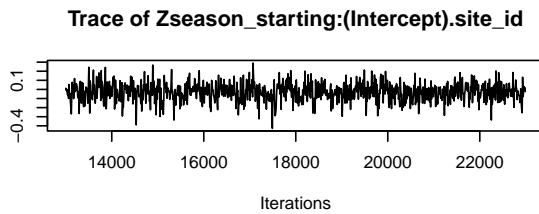
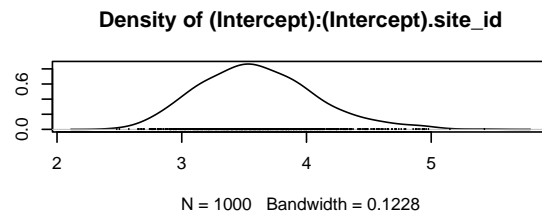
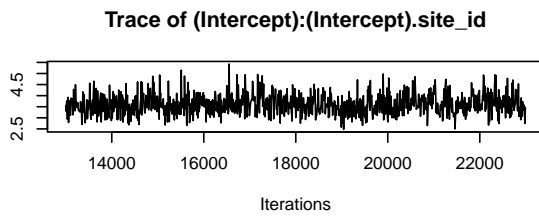
```
##
## Iterations = 13001:22991
## Thinning interval = 10
## Sample size = 1000
##
## DIC: 4793.642
##
## G-structure: ~us(1 + Zseason_starting):site_id
##
##               post.mean l-95% CI u-95% CI eff.samp
## (Intercept):(Intercept).site_id      3.60711   2.7413   4.5256   601.5
## Zseason_starting:(Intercept).site_id  -0.03076  -0.2312   0.1729   661.5
## (Intercept):Zseason_starting.site_id  -0.03076  -0.2312   0.1729   661.5
## Zseason_starting:Zseason_starting.site_id  0.20998   0.1310   0.3008   881.5
##
## R-structure: ~units
##
##      post.mean l-95% CI u-95% CI eff.samp
## units      0.114  0.08879   0.1459    688.7
##
## Location effects: nests ~ Zseason_starting * ZLat
##
##               post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      5.82350  5.48008  6.13709   1000.0 <0.001 ***
## Zseason_starting  -0.18182 -0.27815 -0.07260    826.2  0.002 **
## ZLat              1.42227  1.08402  1.72124   1000.0 <0.001 ***
## Zseason_starting:ZLat  0.04787 -0.04940  0.14896    786.6  0.320
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.2 MCMCglmm diagnostics for mc2

Assessing model convergence. We do this separately for both fixed and random effects. The trace plot should look like a fuzzy caterpillar

```
# plot(mc2$Sol)

# variances of the random effects - shows good mixing
plot(mc2$VCV)
```



```
# what are the effective sample size for the random effects?
coda::effectiveSize(mc2$VCV)
```

```
## (Intercept):(Intercept).site_id
```

```
##                601.4779
##      Zseason_starting:(Intercept).site_id
##                661.5470
##      (Intercept):Zseason_starting.site_id
##                661.5470
## Zseason_starting:Zseason_starting.site_id
##                881.5166
##                units
##                688.7047
```

```
# The effective sample size is large
```

```
# from MCMC Course notes (page 60):
```

```
diag(autocorr(mc2$VCV)[2, , ]) # low autocorrelation
```

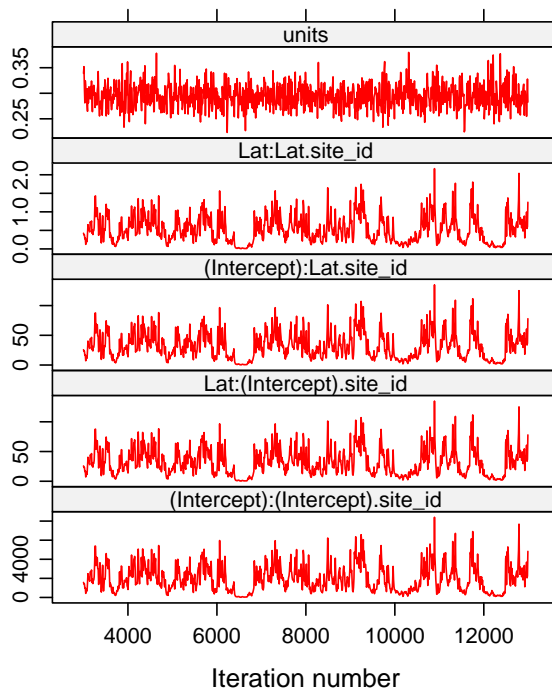
```
##      (Intercept):(Intercept).site_id
##                0.01987222
##      Zseason_starting:(Intercept).site_id
##                0.06935291
##      (Intercept):Zseason_starting.site_id
##                0.06935291
## Zseason_starting:Zseason_starting.site_id
##                0.06247350
##                units
##                0.18385592
```

```
# the variance components
```

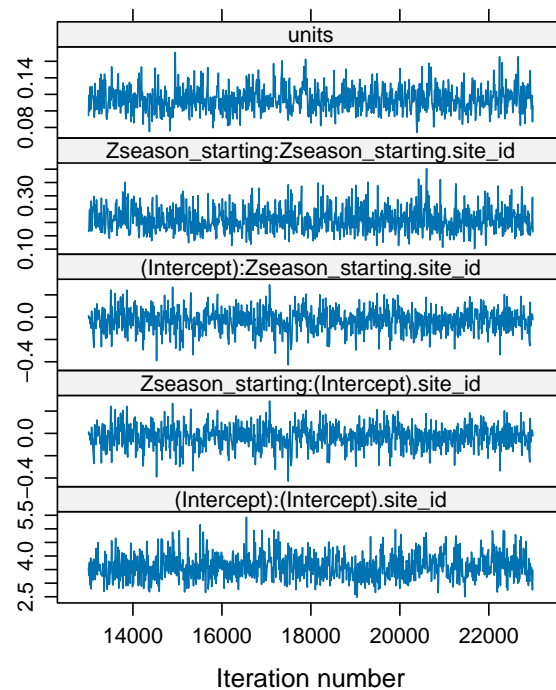
```
trace2 = lattice::xyplot(as.mcmc(mc2$VCV), par.strip.text=list(cex=0.8))
```

```
cowplot::plot_grid(traceK, trace2, labels = c('Kruger (2023)', 'Oosthuizen et al'), ncol = 2, label_size = 12)
```

Kruger (2023)



Oosthuizen et al



```
# Save Plot
# pdf("./figures/Supp_Trace_plots.pdf",
#      useDingbats = FALSE, width = 8, height = 12)
# cowplot::plot_grid(traceK, trace2, labels = c('A', 'B'),
#                     ncol = 2, label_size = 14, vjust = 3.5, hjust = -.5)
# dev.off()
```

3.3 Predict using MCMCglmm mc2

```
# construct an hypothetical dataframe to predict to
```

```
# need to predict to z-standardized variables
Z1 = dplyr::select(nestM3, season_starting, Lat)
Z2 <- scale(Z1)
attr(Z2, "scaled:center")
```

```
## season_starting      Lat
##      2003.32985      -62.96523
```

```
attr(Z2, "scaled:scale")
```

```
## season_starting      Lat
##      15.685180      1.600454
```



```

ave_ss = attr(Z2,"scaled:center")[[1]]
ave_lat = attr(Z2,"scaled:center")[[2]]

sd_ss = attr(Z2,"scaled:scale")[[1]]
sd_lat = attr(Z2,"scaled:scale")[[2]]

years<-data.frame(season_starting=c(1960:2020)) # extrapolate to 1960
#years<-data.frame(season_starting=c(1980:2020)) # extrapolate to 1980

pops<-data.frame(site_id=countsN$site_id[countsN$ncounts>1],
                 Lat=countsN$Lat[countsN$ncounts>1])
popy<-merge(pops,years)
popy$nested<-c(0) ### MCMCglmm needs a column with the response variable

popy$Zseason_starting = (popy$season_starting - ave_ss)/sd_ss
popy$ZLat = (popy$Lat - ave_lat)/sd_lat

head(popy)

```

```

##   site_id   Lat season_starting nests Zseason_starting   ZLat
## 1  AILS -60.780          1960     0      -2.762471  1.3653784
## 2  AITC -62.407          1960     0      -2.762471  0.3487919
## 3  AITK -60.738          1960     0      -2.762471  1.3916209
## 4  ALCO -64.240          1960     0      -2.762471 -0.7965080
## 5  AMPH -60.684          1960     0      -2.762471  1.4253614
## 6  ANDE -60.757          1960     0      -2.762471  1.3797493

```

```

# Don't extrapolate more than X years
first_last_season = nestM3 %>%
  dplyr::group_by(site_id) %>%
  dplyr::summarise(minyear = min(season_starting),
                  maxyear = max(season_starting)) %>%
  dplyr::arrange(minyear)
first_last_season

```

```

## # A tibble: 146 x 3
##   site_id minyear maxyear
##   <chr>   <int>   <int>
## 1 DEEI    1965    1965
## 2 ORRO    1966    2019
## 3 BRIT    1970    2016
## 4 CCMP    1970    2019
## 5 CLIN    1970    2019
## 6 CRAB    1970    2019
## 7 DECI    1970    2019
## 8 EIP1    1970    2019
## 9 EIP2    1970    2019
## 10 EIP3   1970    2019
## # i 136 more rows

```

```

popy = merge(popy, first_last_season)

```

```
length(unique(popy$site_id))
```

```
## [1] 146
```

```
popypred <- data.frame(predict(mc2,
                              newdata=popy,
                              type="response",
                              marginal=NULL,      # crucial, and not default code.
                              interval="prediction",
                              posterior="all"))
```

```
head(popypred)
```

```
##           fit   lwr   upr
## 1 10359.383 1647 24293
## 2  5805.399 1547 11399
## 3  9246.391 1375 21888
## 4  8244.895 1424 17783
## 5  7463.765 1843 15006
## 6  4439.312 1381  8268
```

```
popy$Zfit = popypred$fit
popy$Zlwr = popypred$lwr
popy$Zupr = popypred$upr
```

```
## How accurate are the predictions relative to observed data?
```

3.4 Conditional model predictions

Plot the observed data against predicted data, per site, to see whether observed data and predicted data agree for the revised analysis

```
required_n_pages = round(133/16)+1

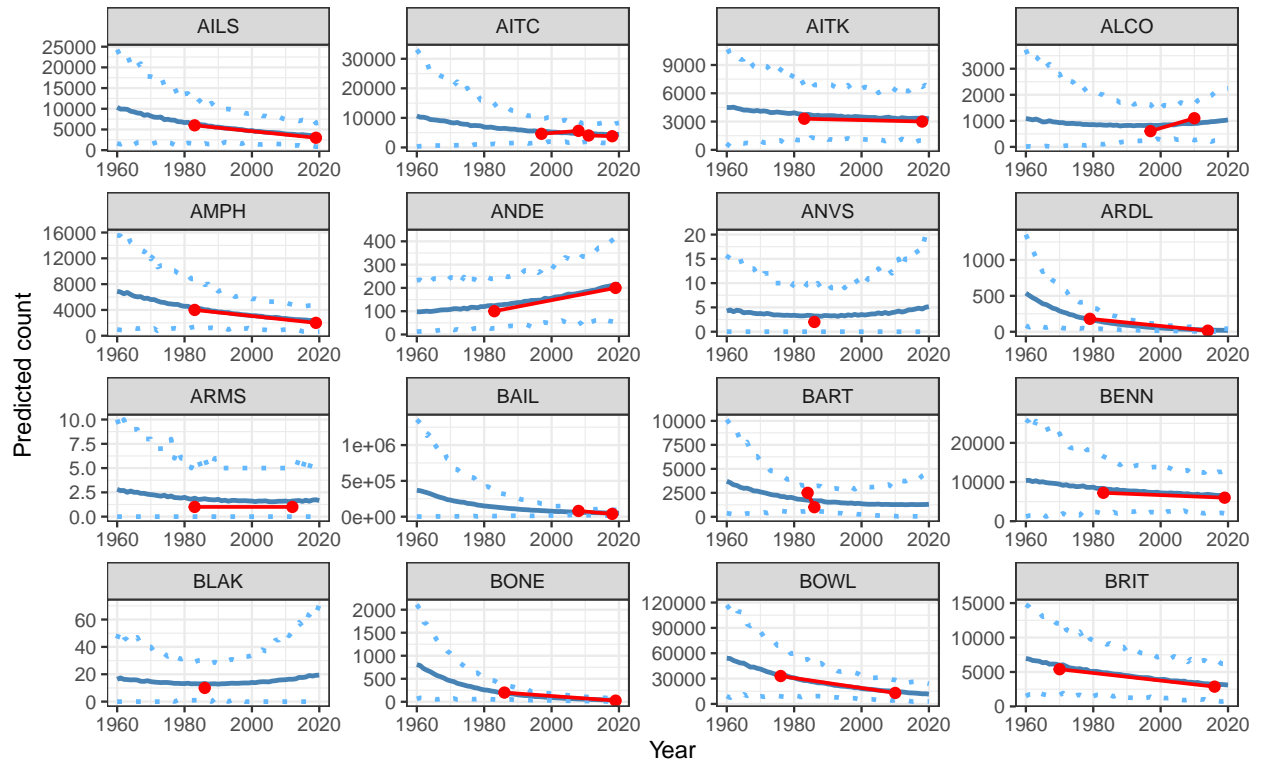
for(i in 1:required_n_pages){

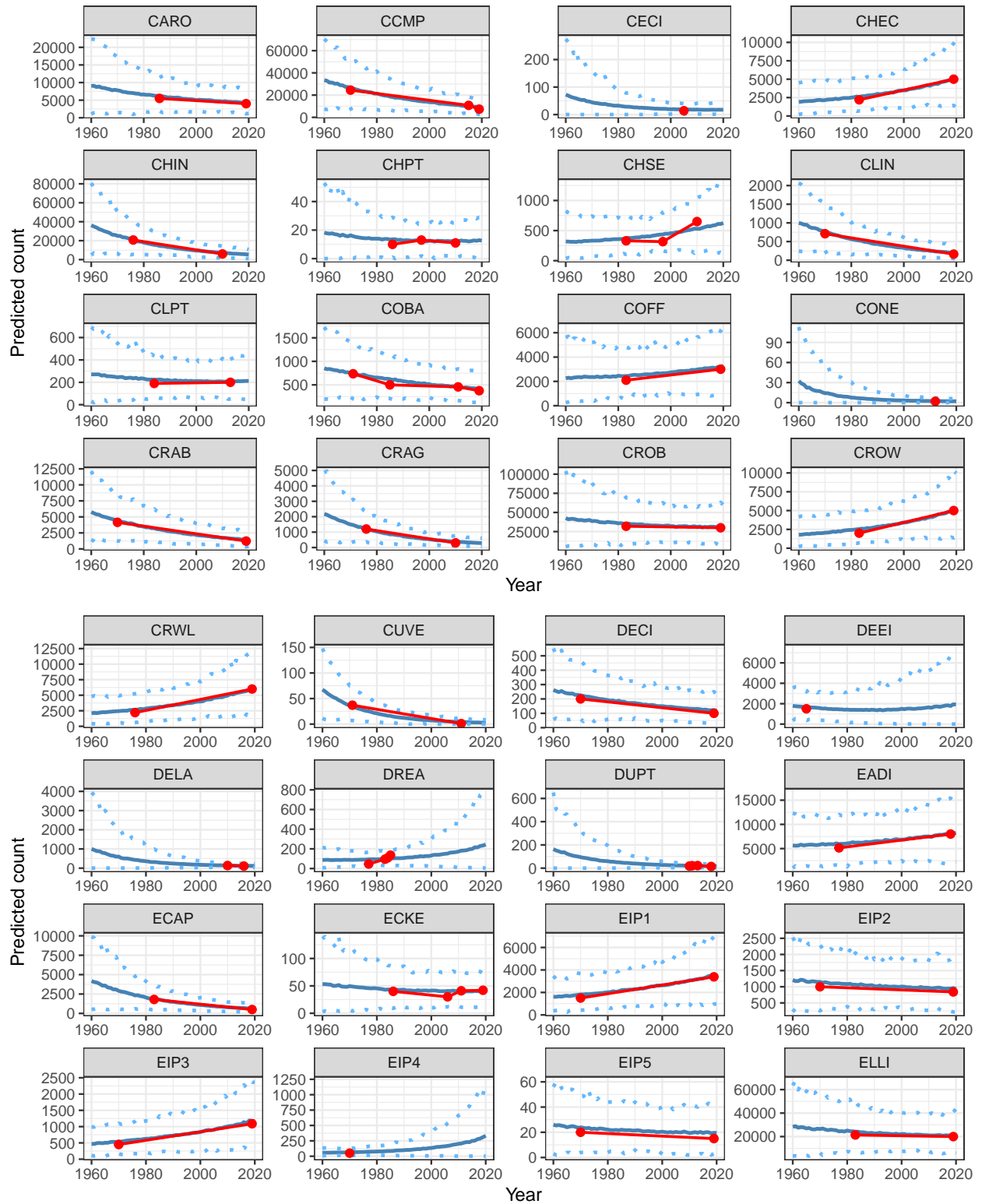
print(ggplot(data = popy) +
      geom_line(aes(x = season_starting, y = Zfit),
                col = "steelblue", linewidth=1.04) +
      geom_line(aes(x = season_starting, y = Zlwr),
                col = "steelblue1", linetype="dotted", linewidth = 1.02) +
      geom_line(aes(x = season_starting, y = Zupr),
                col = "steelblue1", linetype="dotted", linewidth=1.02) +
      geom_point(data = nestm3, aes(season_starting, y = nests),
                 color = "red", cex = 2) +
      geom_line(data = nestm3, aes(season_starting, y = nests),
                 color = "red", linewidth=0.8) +
      theme_bw() +
      xlab("Year") +
```

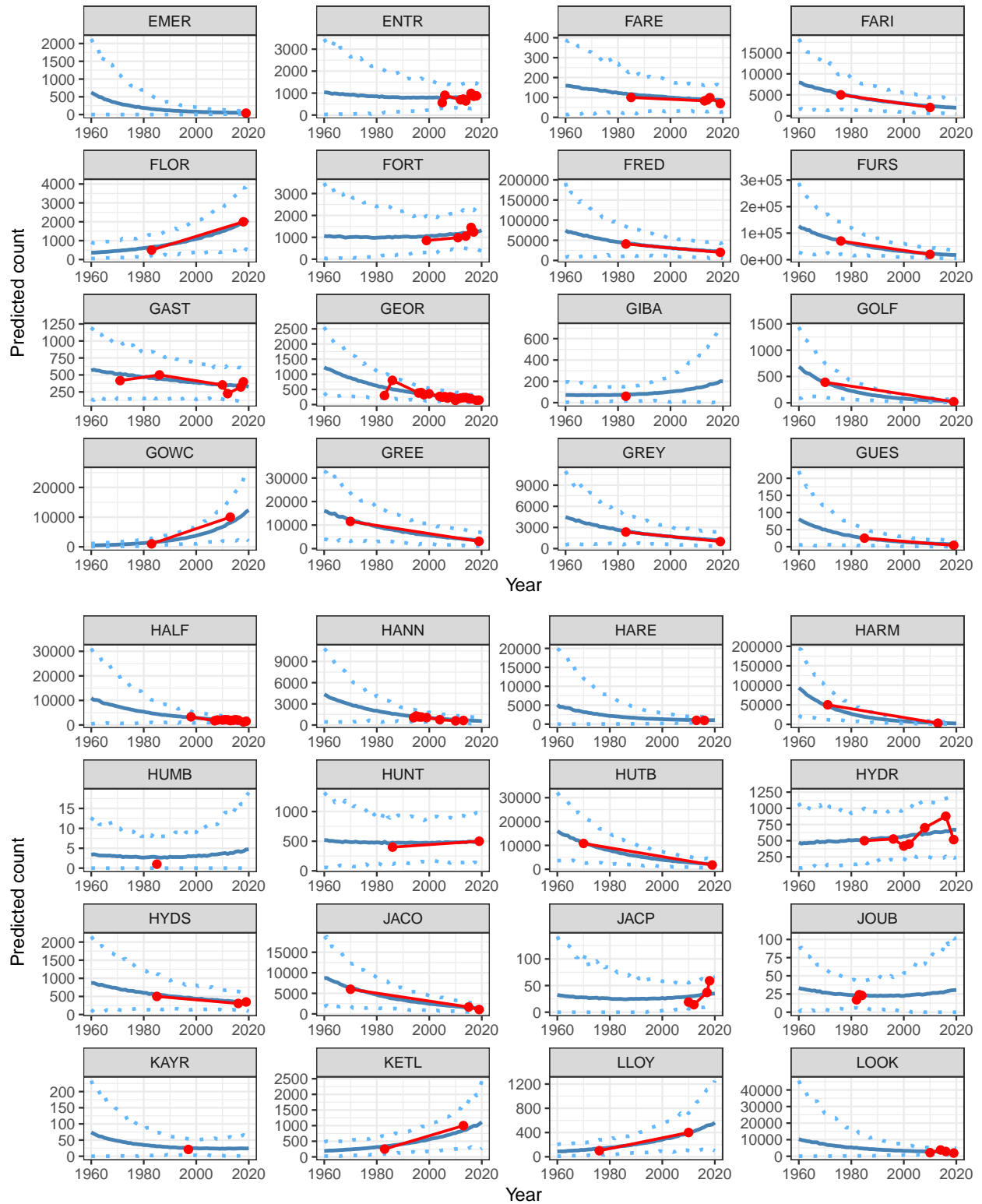
```

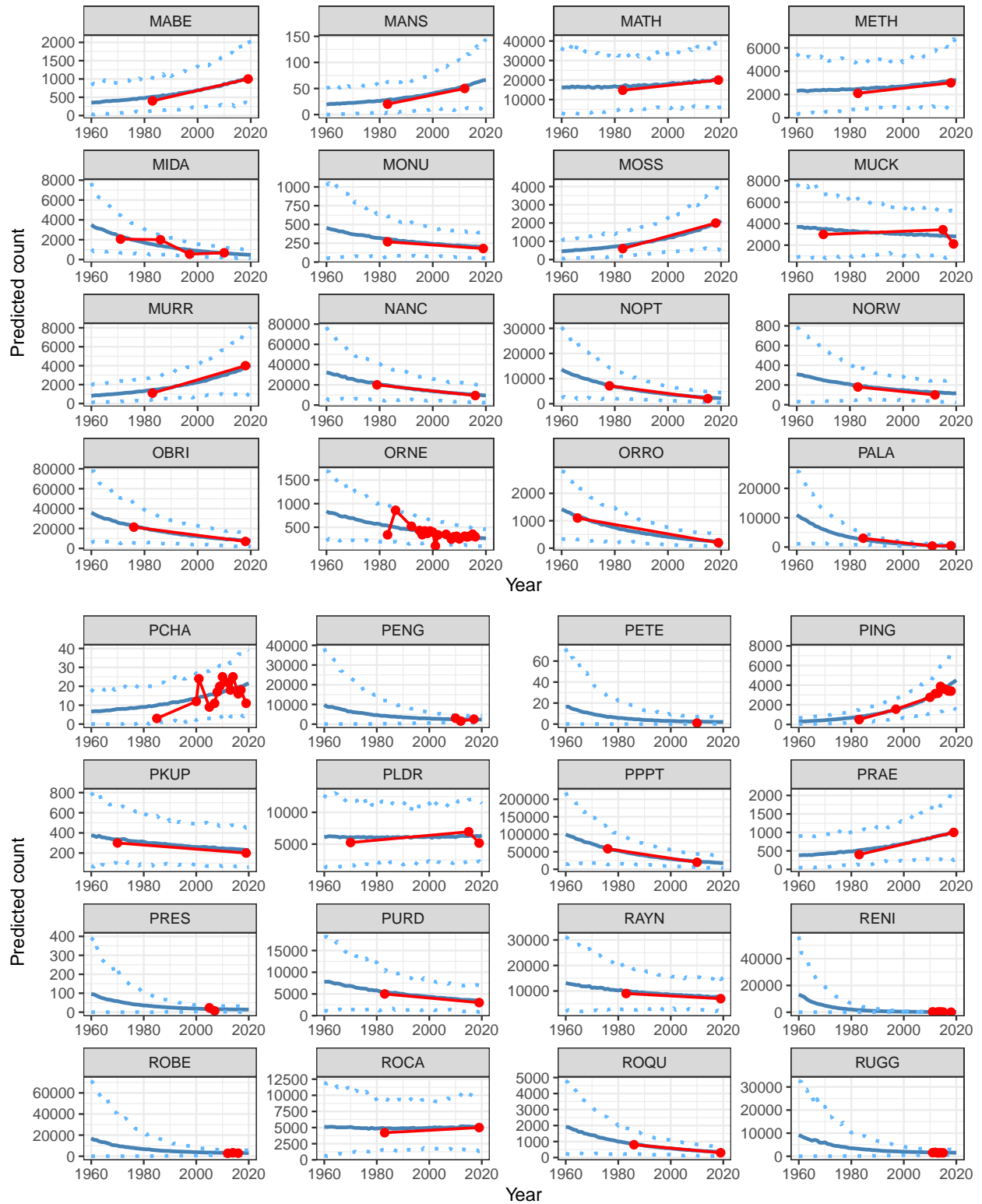
ylab("Predicted count") +
# theme(strip.text = element_text(size = 1.5)) +
facet_wrap_paginate(~ site_id, ncol = 4, nrow = 4,
                    page = i,
                    scales = 'free'))}

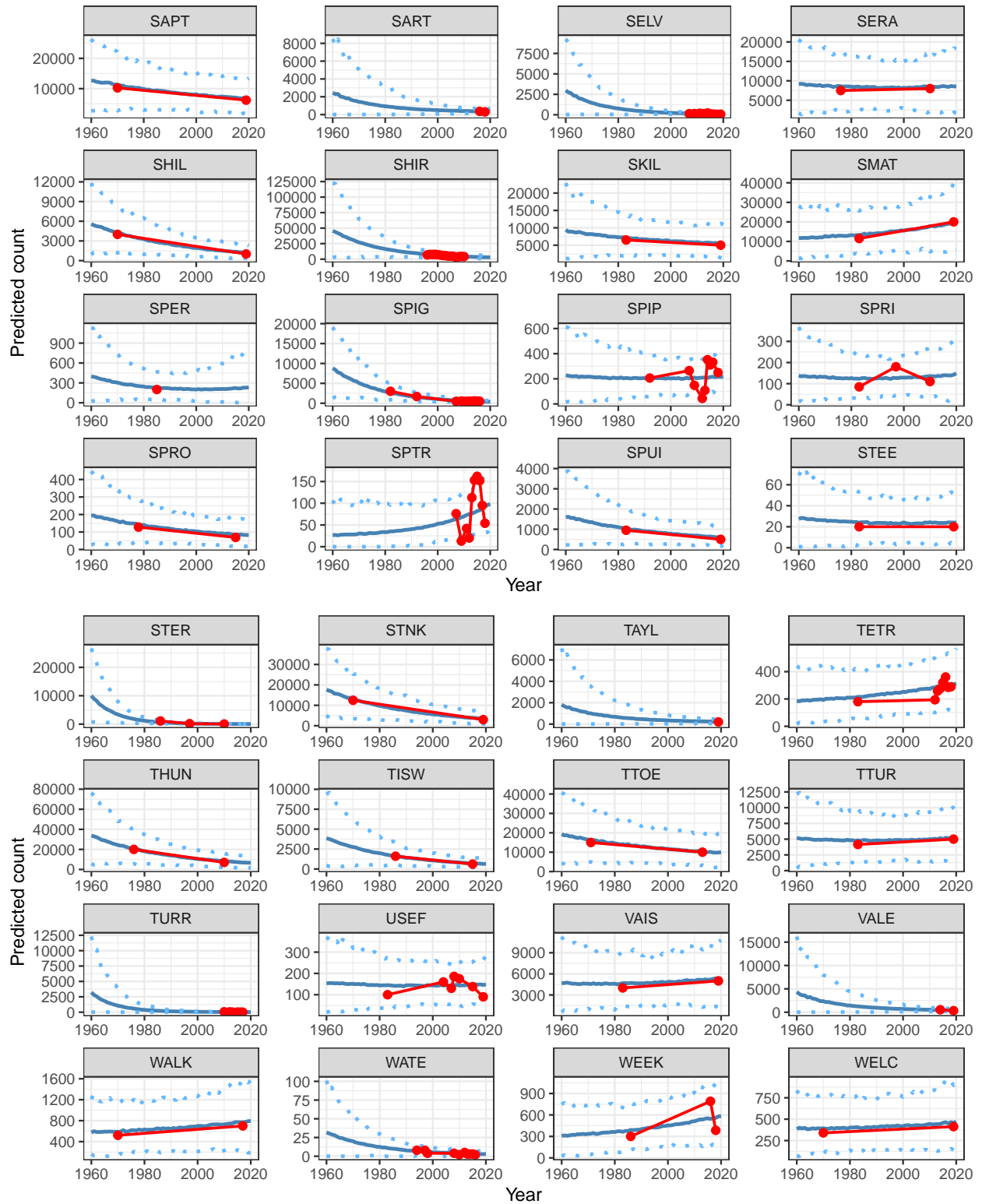
```











Predictions are good, although back-predicting to 1960 is extrapolation
 # (there are only 2 counts prior to 1970) so uncertainty (prediction intervals)
 # is high.

3.5 Marginal model predictions

```
# This predicts the population average response - i.e., a similar prediction at all sites.
# If you are adding up the individual sites, as Kruger 2023 did to calculate popT,
# then you are adding predictions of the average response every time.
popypred_marg <- data.frame(predict(mc2,
                                newdata=popy,
                                type="response",
                                #marginal=NULL,          # crucial, and not default code.
                                marginal=~us(1 + Zseason_starting):site_id,
                                interval="prediction",
                                posterior="all"))

popy$Zfit_marg = popypred_marg$fit
popy$Zlwr_marg = popypred_marg$lwr
popy$Zupr_marg = popypred_marg$upr

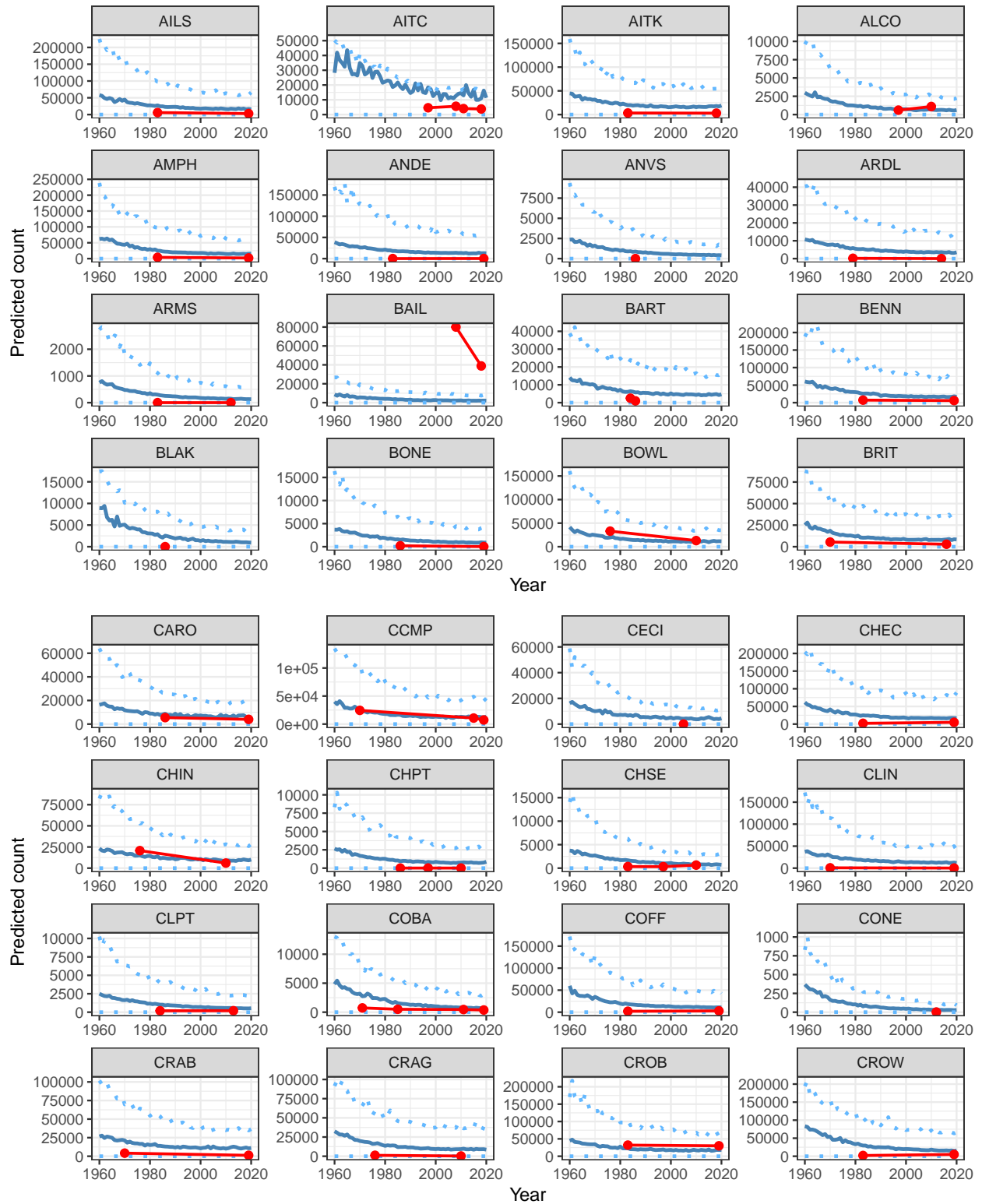
## How accurate are the predictions relative to observed data?

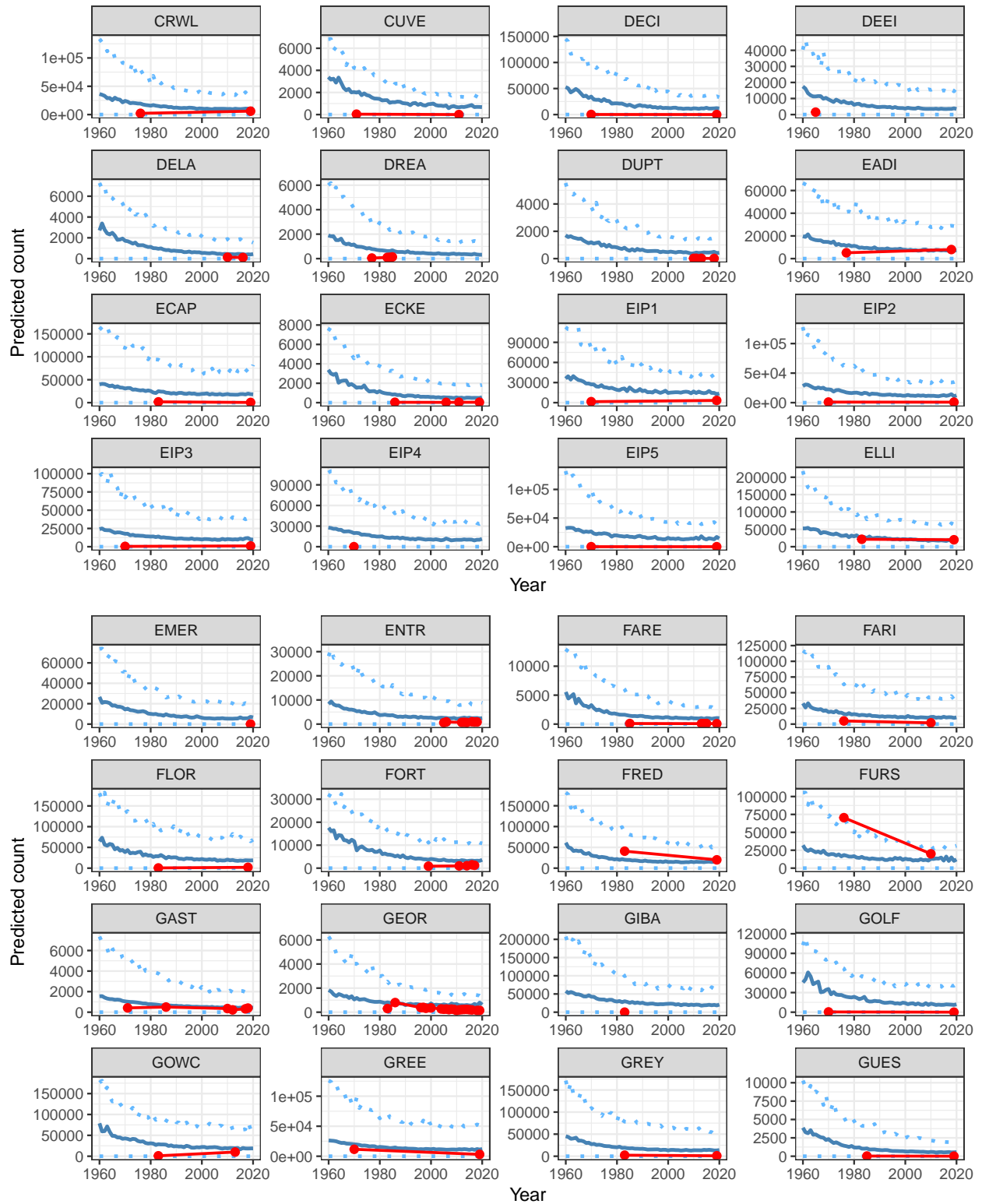
required_n_pages = round(133/16)+1

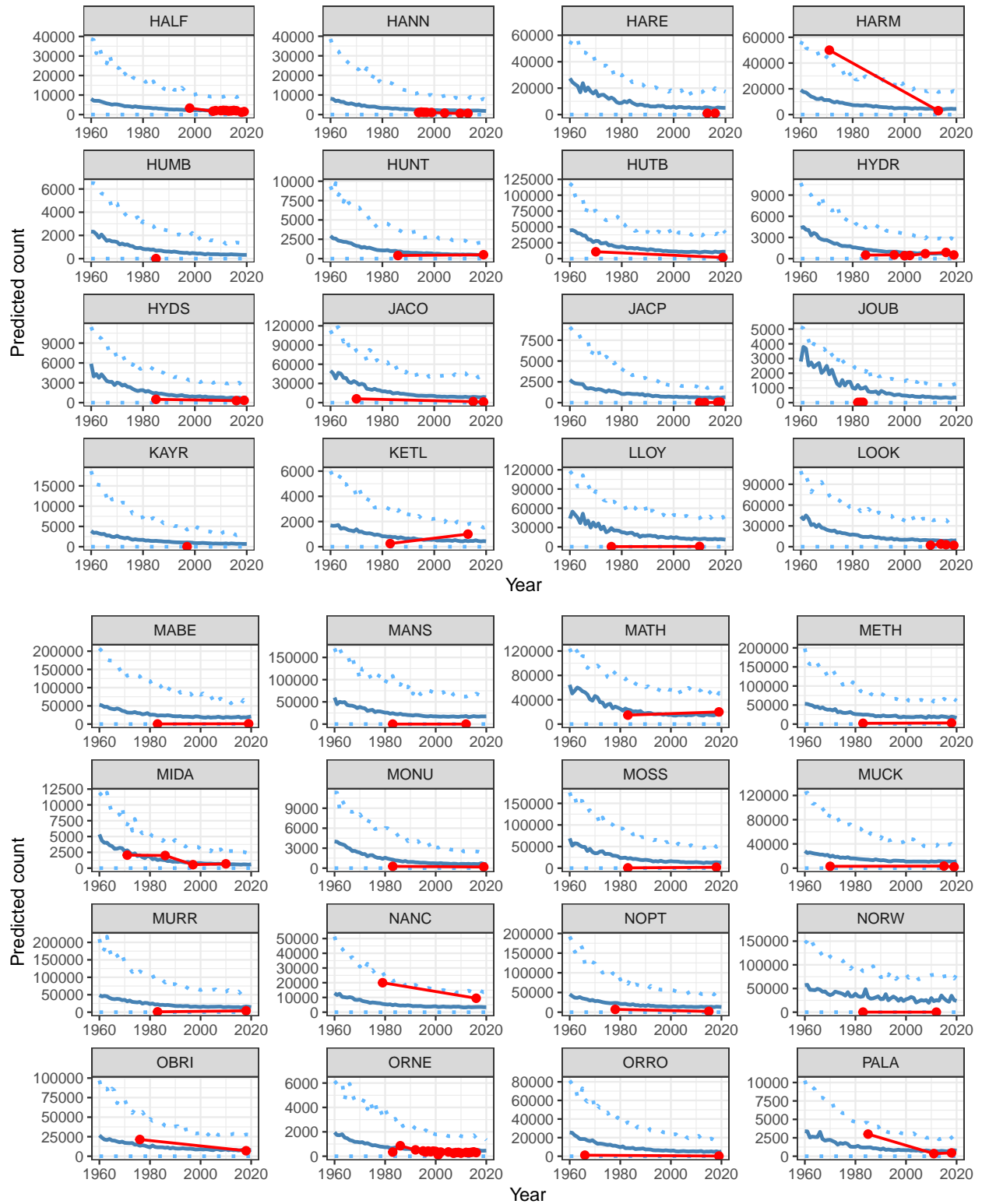
for(i in 1:required_n_pages){

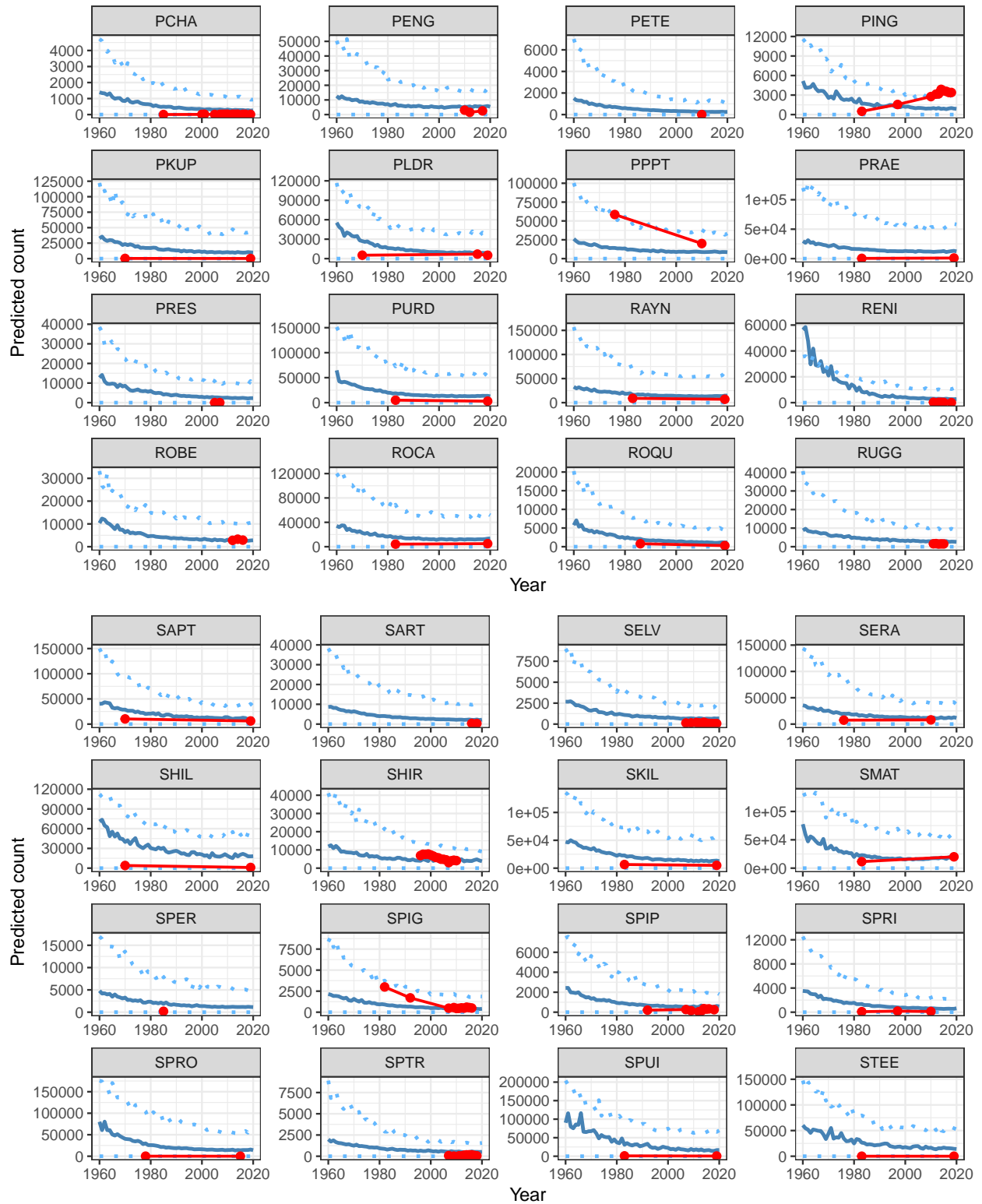
print(ggplot(data = popy) +
      geom_line(aes(x = season_starting, y = Zfit_marg),
                col = "steelblue", linewidth=1.04) +
      geom_line(aes(x = season_starting, y = Zlwr_marg),
                col = "steelblue1", linetype="dotted", linewidth = 1.02) +
      geom_line(aes(x = season_starting, y = Zupr_marg),
                col = "steelblue1", linetype="dotted", linewidth=1.02) +
      geom_point(data = nestm3, aes(season_starting, y = nests),
                 color = "red", cex = 2) +
      geom_line(data = nestm3, aes(season_starting, y = nests),
                 color = "red",size=0.8) +
      theme_bw() +
      xlab("Year") +
      ylab("Predicted count") +
      # theme(strip.text = element_text(size = 1.5)) +
      facet_wrap_paginate(~ site_id, ncol = 4, nrow = 4,
                          page = i,
                          scales = 'free'))}

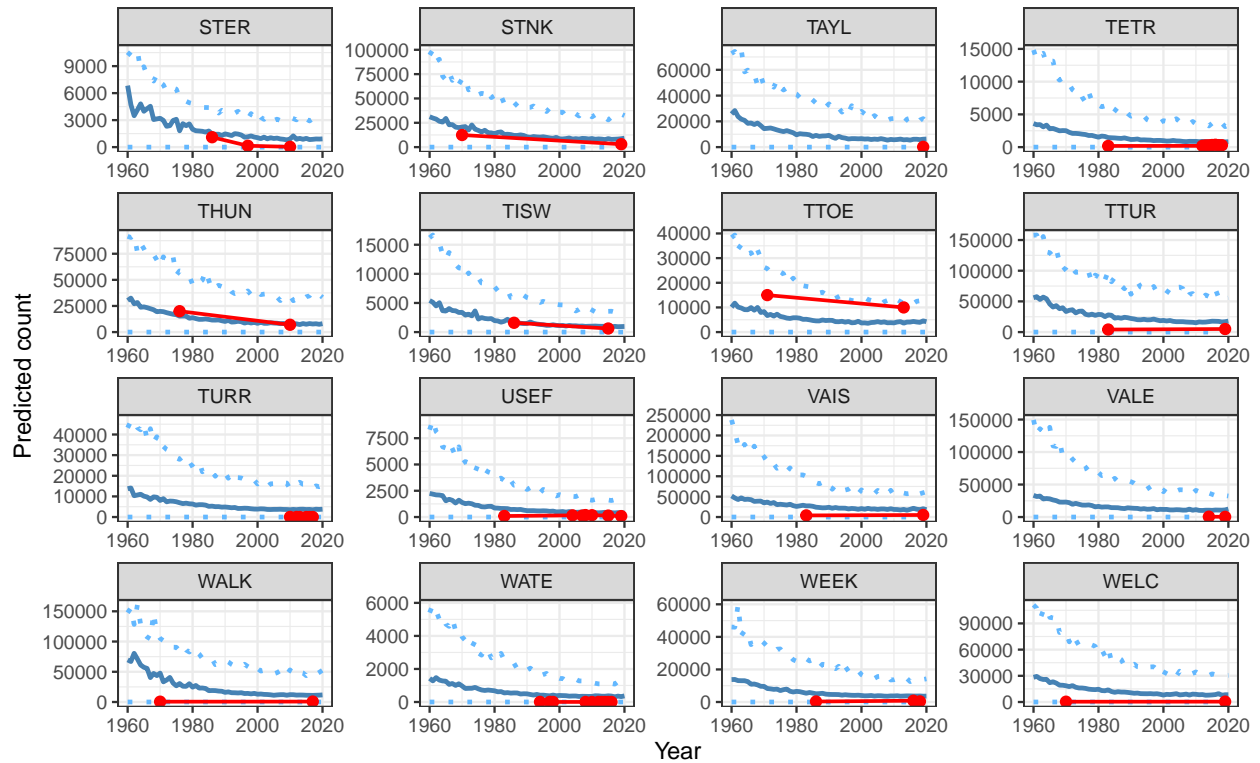
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```









3.6 Revised plots of latitude and slope (population change)

```
# extract random effects from MCMCglmm
# https://stackoverflow.com/questions/64562052/extract-random-effects-from-mcmcglmm
library(broom.mixed)
re = tidy(mc2, effects="ran_vals")
unique(re$group)
```

```
## [1] "site_id"
```

```
re = re %>%
  dplyr::select(-group, -effect) %>%
  pivot_wider(names_from = term, values_from = c(estimate, std.error))

head(re)
```

```
## # A tibble: 6 x 5
##   level 'estimate_(Intercept)' estimate_Zseason_starting 'std.error_(Intercept)'
##   <chr>                <dbl>                <dbl>                <dbl>
## 1 AILS                  0.530                -0.143               0.323
## 2 AITC                  2.12                 0.0206              0.243
## 3 AITK                  0.254                0.0649              0.331
## 4 ALCO                  1.97                 0.330               0.336
## 5 AMPH                  0.0518              -0.156               0.332
## 6 ANDE                 -2.76                0.361               0.327
## # i 1 more variable: std.error_Zseason_starting <dbl>
```

```

# estimate_(Intercept) is related to the initial population size
# estimate_Zseason_starting is the slope of population increase (+)
# or decrease (-)

names(re) = c("site_id", "est_int", "estZss",
              "se_int", "seZss")

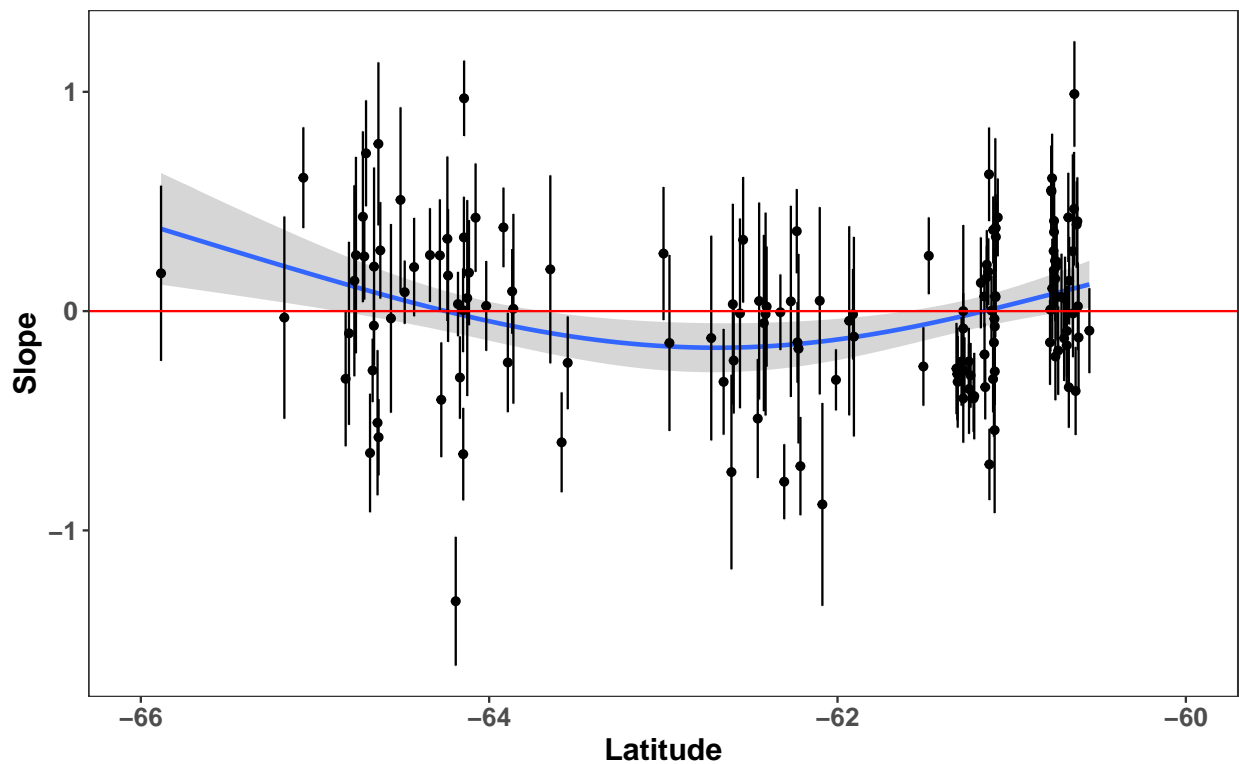
# add latitude
nestM3_lat = dplyr::select(nestM3, Lat, site_id) %>%
  dplyr::distinct(site_id, Lat)

re = left_join(re, nestM3_lat, by = "site_id")

# plot relationship between slope and latitude

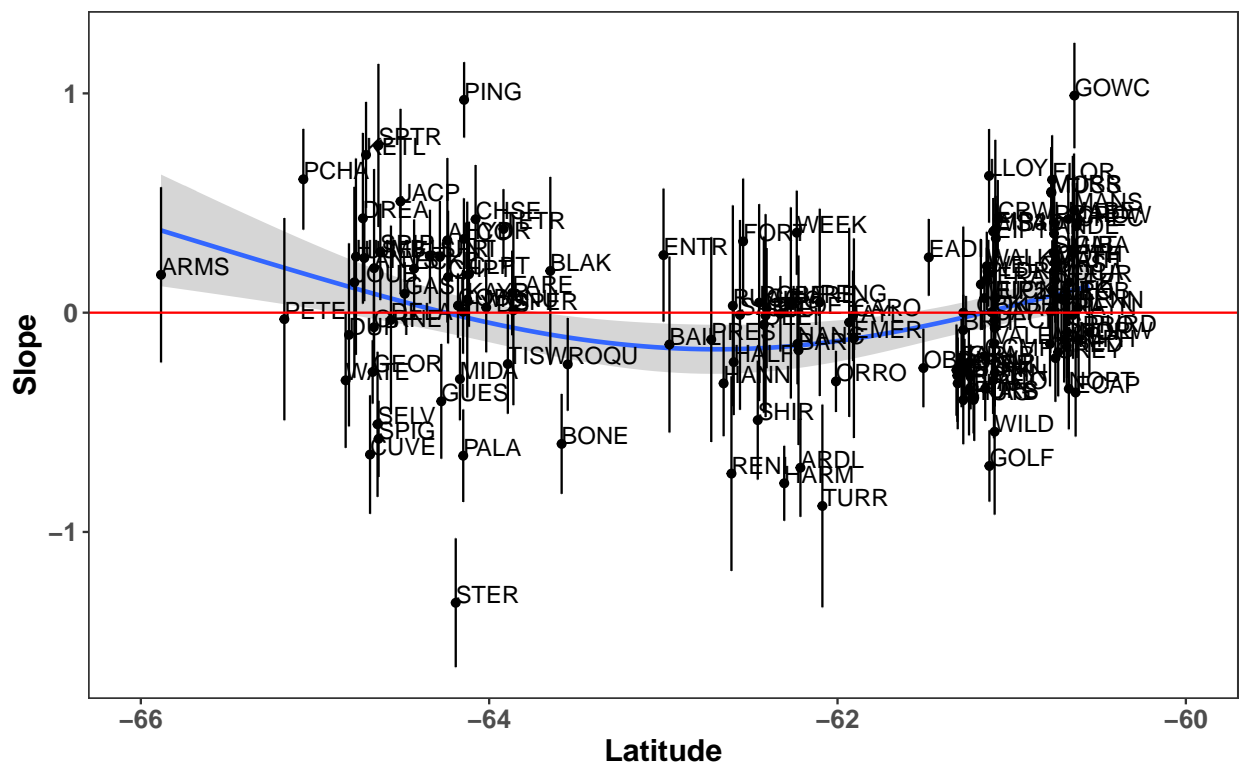
ggplot(data = re, aes(x = Lat, y = estZss))+
  stat_smooth(method="gam", formula=y~s(x,k=2))+
  # geom_smooth(method='lm', formula= y~x)+
  geom_point()+
  geom_errorbar(aes(ymin=estZss-seZss,
                   ymax=estZss+seZss))+
  theme_bw()+th+
  ylab("Slope")+xlim(-66,-60)+
  xlab("Latitude")+
  geom_hline(yintercept=0,
            color = "red")

```



```
ggplot(data = re, aes(x = Lat, y = estZss, label = site_id))+
  stat_smooth(method="gam", formula=y~s(x,k=2))+
  # geom_smooth(method='lm', formula= y~x)+
  geom_point()+
  geom_text(hjust=0, vjust=0) +
  geom_errorbar(aes(ymin=estZss-seZss,
                    ymax=estZss+seZss))+

  theme_bw()+th+
  ylab("Slope")+xlim(-66,-60)+
  xlab("Latitude")+
  geom_hline(yintercept=0,
            color = "red")
```



The following figures plot the data from the original study

4 Oosthuizen et al data distribution figures

```
# This shows that there are some 1-count sites in the data being analysed
# (n = 146, not n = 133)
samplesize = nestM3 %>% group_by(site_id, ncounts) %>% tally()
length(unique(nestM3$site_id))
```

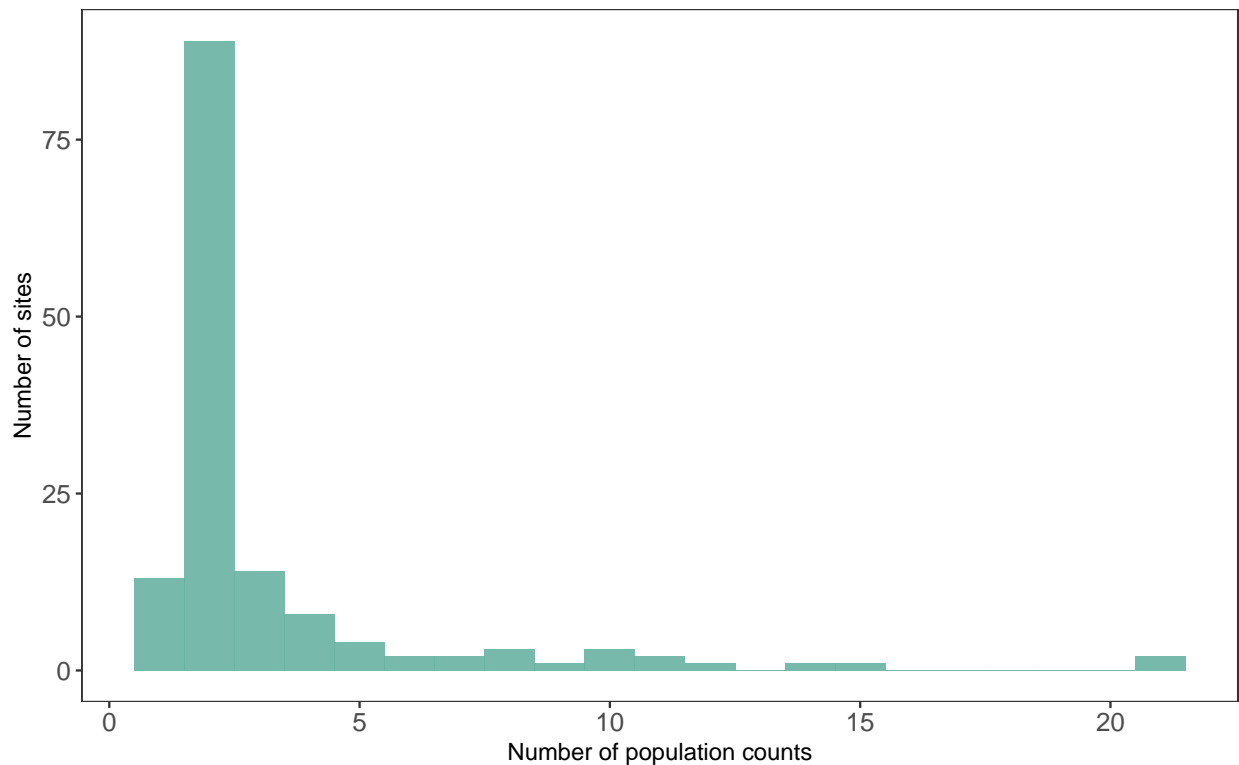
```
## [1] 146
```

```

samplesize.plot <- samplesize %>%
  ggplot(aes(x=n)) +
  geom_histogram(binwidth=1, fill="#69b3a2", alpha=0.9) +
  theme_bw() +
  ylab("Number of sites") +
  xlab("Number of population counts") +
  theme(axis.text=element_text(size=12),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())

samplesize.plot

```



```

## Save Plot
# pdf("./Figure samplesize.pdf",
#     useDingbats = FALSE, width = 4, height = 4)
# samplesize.plot
# dev.off()

samplesizeYear = nestM3 %>% group_by(season_starting) %>% tally()
#samplesizeYear

samplesizeYear.plot = samplesizeYear %>%
  ggplot(aes(x=season_starting, y = n)) +
  geom_bar(stat = "identity", fill="#69b3a2", alpha=0.9) +
  theme_bw() +
  ylab("Number of sites counted") +
  xlab("Year") +
  theme(axis.text=element_text(size=12),

```

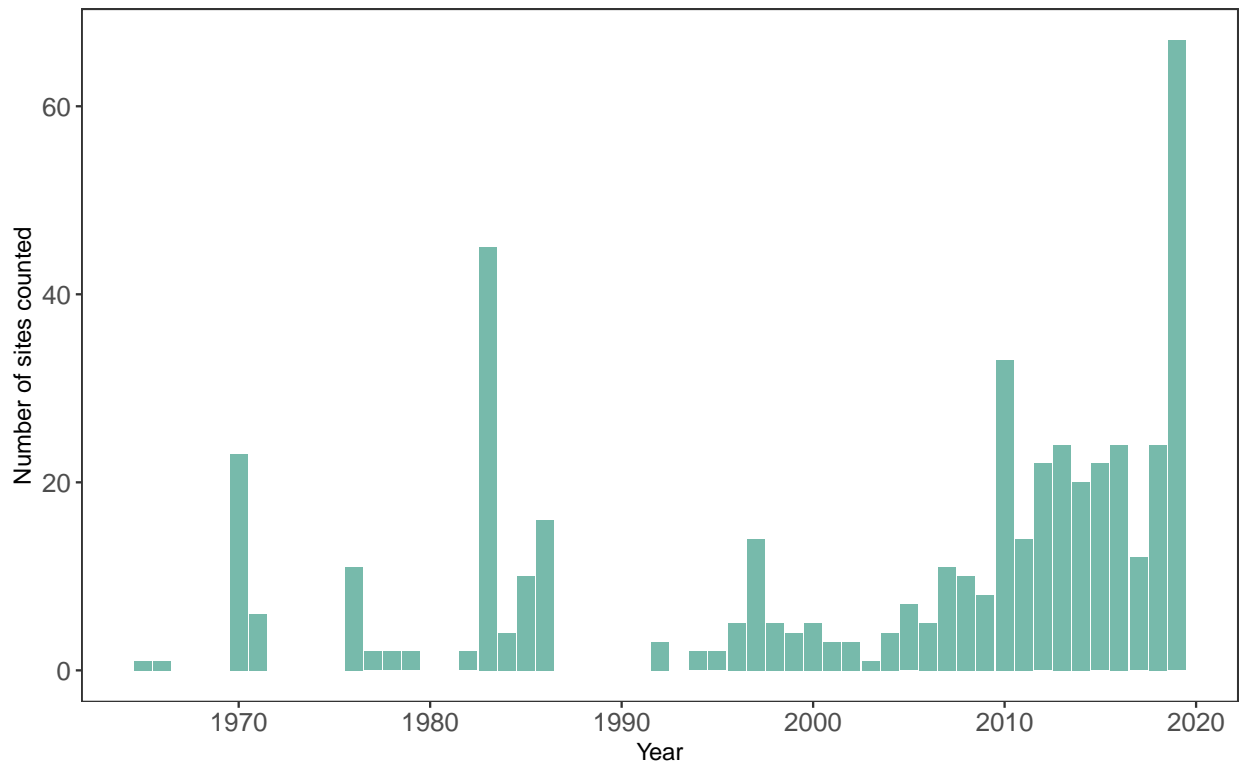


```

    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()+
    scale_x_continuous(breaks = seq(1960, 2020, by = 10))

```

`samplesizeYear.plot`



```

## Save Plot
# pdf("./Figure samplesizeYear.pdf",
#     useDingbats = FALSE, width = 6, height = 4)
# samplesizeYear.plot
# dev.off()

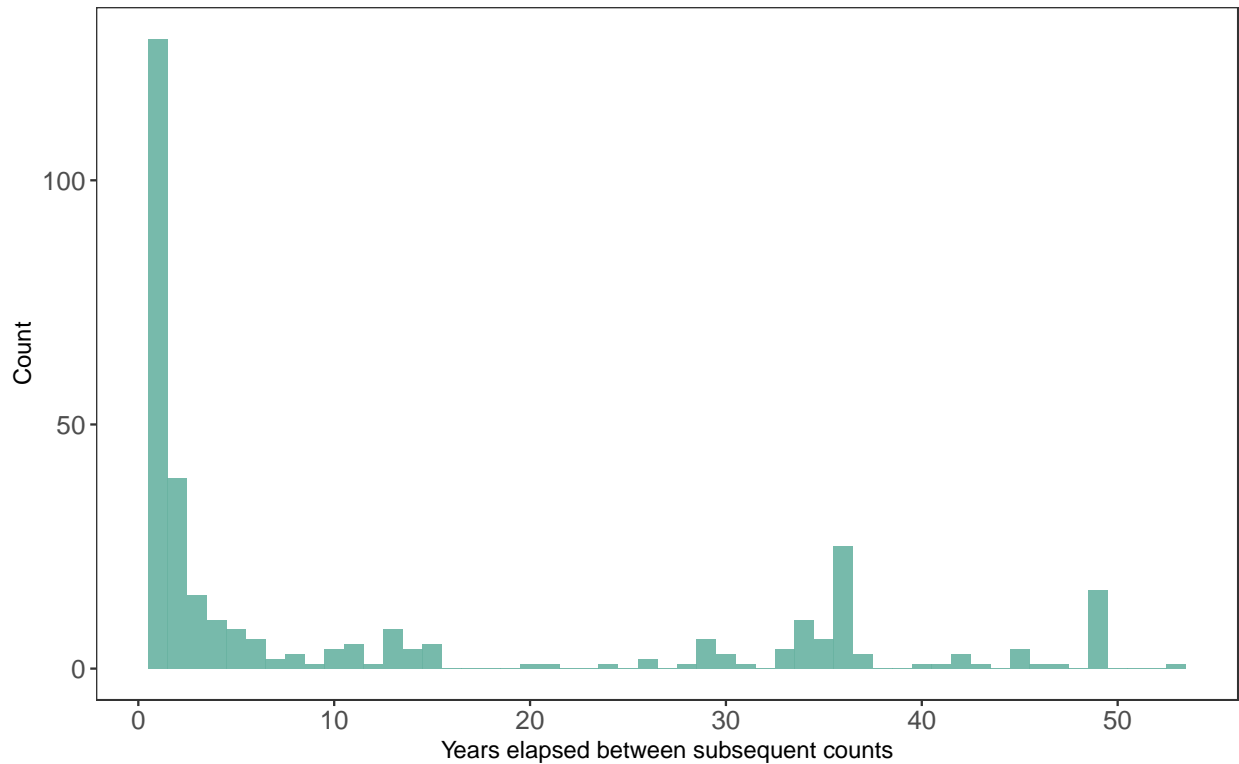
# time between counts per site
diff = nestm3 %>%
  dplyr::arrange(site_id, season_starting) %>%
  dplyr::group_by(site_id) %>%
  dplyr::mutate(time.difference = season_starting - lag(season_starting))
#diff

diff.plot = diff %>%
  ggplot(aes(x=time.difference)) +
  geom_histogram(binwidth=1, fill="#69b3a2", alpha=0.9) +
  theme_bw()+
  ylab("Count")+
  xlab("Years elapsed between subsequent counts") +
  theme(axis.text=element_text(size=12),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+

```

```
scale_x_continuous(breaks = seq(0, 50, by = 10))
```

```
diff.plot
```



```
## Save Plot
# pdf("../Figure timedifference.pdf",
#       useDingbats = FALSE, width = 6, height = 4)
# diff.plot
# dev.off()

library(colorspace)
library(scales)

nestm3$countbreaks = cut(nestm3$ncounts, c(0, 2, 3, 5, 9, Inf))

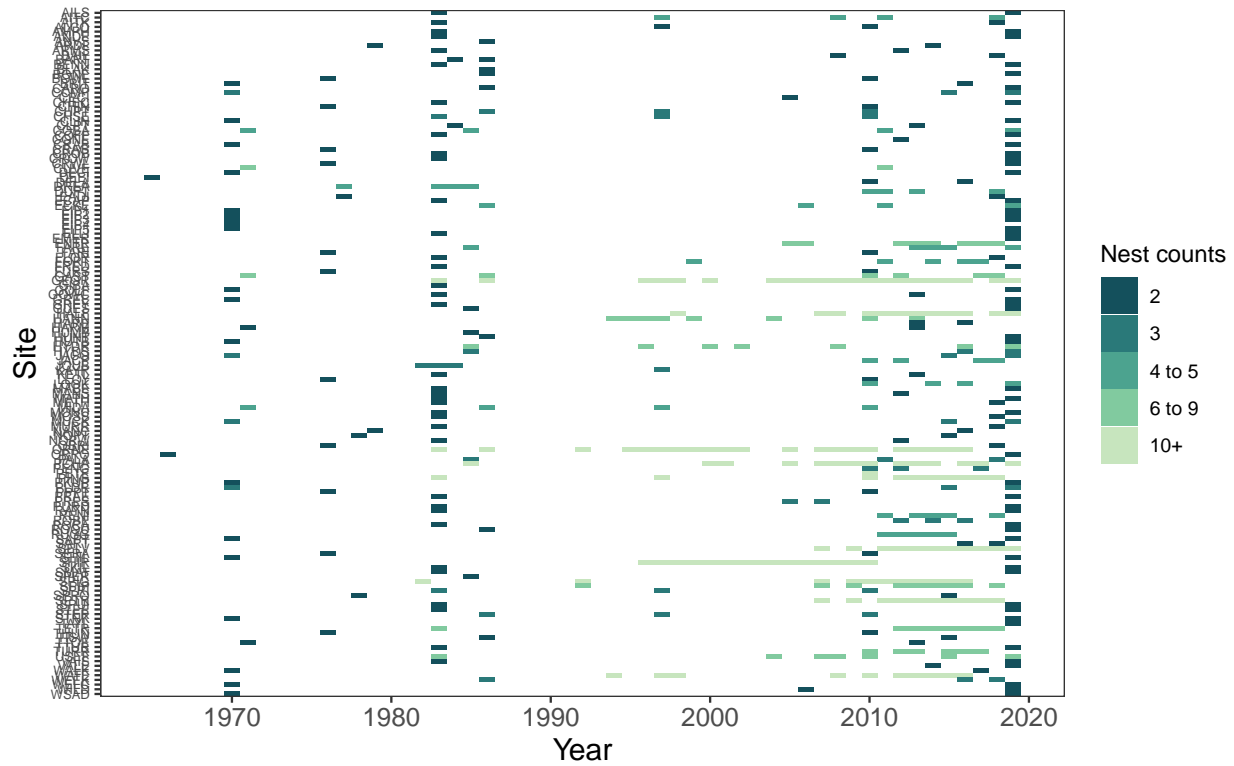
heat = ggplot(nestm3, aes(x = as.numeric(season_starting),
                          y = site_id,
                          fill= cut(ncounts, c(0, 2, 3, 5, 9, Inf),
                                     labels = c('2', '3', '4 to 5', '6 to 9', '10+')))) +
  geom_tile() +
  scale_fill_discrete_sequential(palette = "BluGrn", rev = F) +
  guides(fill=guide_legend(title="Nest counts")) +
  theme_bw() +
  ylab("Site") +
  xlab("Year") +
  theme(axis.text.x=element_text(size=12),
        axis.title.x=element_text(size=14),
        axis.text.y = element_text(size = 6),
```

```

axis.title.y=element_text(size=14),
panel.grid.major = element_blank(),
panel.grid.minor = element_blank())+
scale_x_continuous(breaks = seq(1960, 2020, by = 10))+
scale_y_discrete(limits=rev)

```

heat

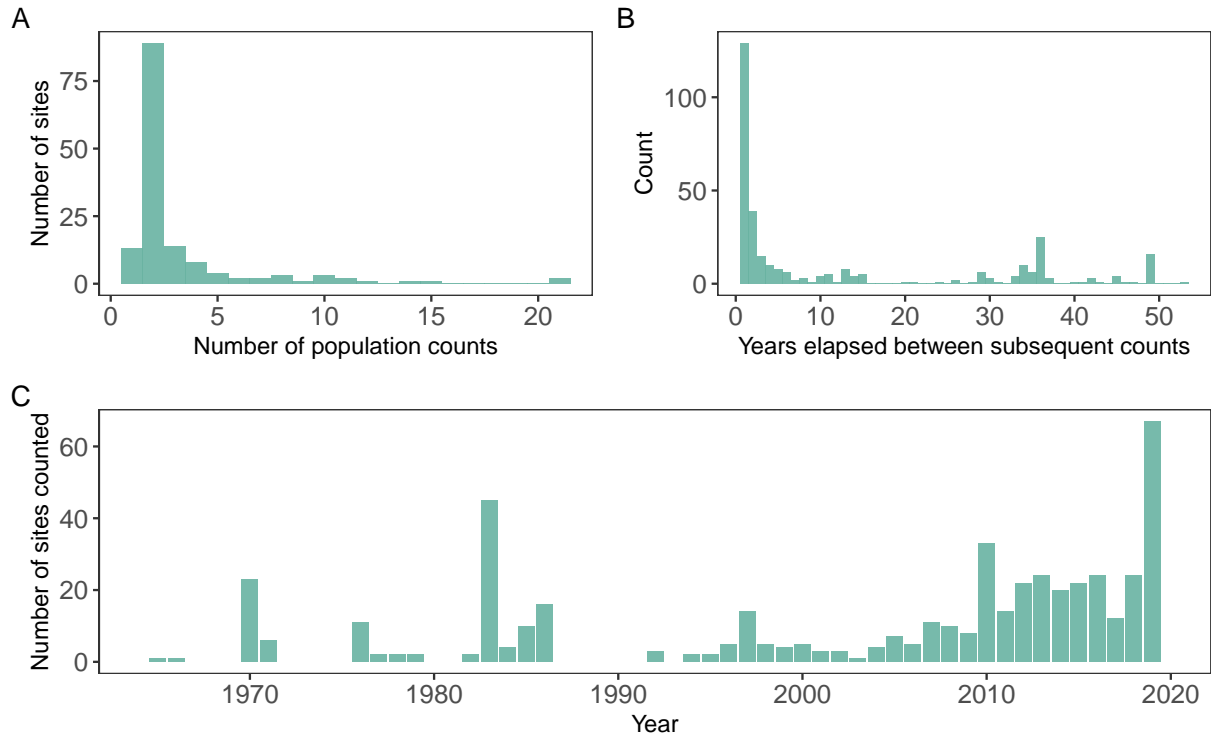


```

## Save Plot
# pdf("./Figure samplesize_heat.pdf",
#     useDingbats = FALSE, width = 7, height = 8)
# heat
# dev.off()

library(patchwork)
combinedfig = (samplesize.plot | diff.plot) / samplesizeYear.plot +
  plot_layout(nrow = 2, widths = c(1, 3)) +
  plot_annotation(tag_levels = 'A')
combinedfig

```



```
## Save Plot
# pdf("../figures/Figure combined.pdf",
#       useDingbats = FALSE, width = 8, height = 6)
# combinedfig
# dev.off()
```

5 Oosthuizen et al - population change:

How many penguins were there, per year, across all sites? We don't know this from counts, as we only have intermittent counts. Estimate and plot the total population size predicted per year (how many penguins were there in all populations?)

```
head(popy)
```

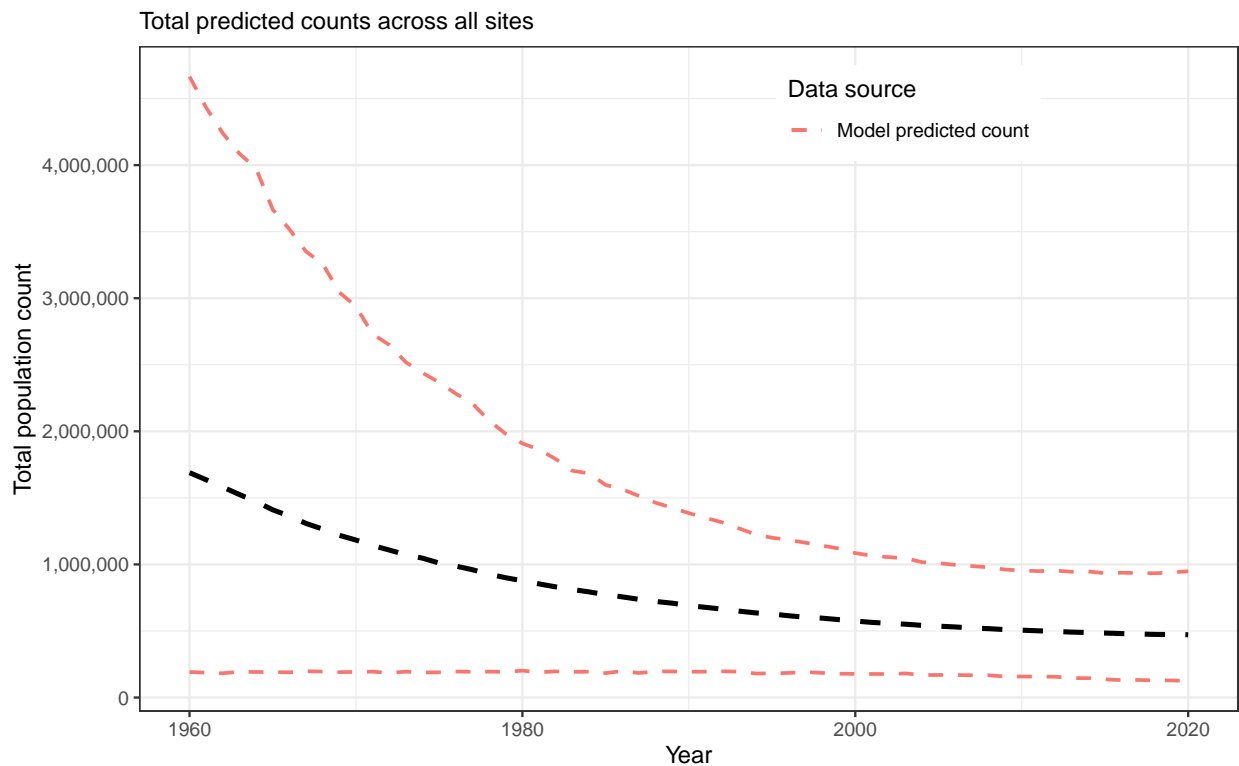
```
##   site_id   Lat season_starting nests Zseason_starting   ZLat minyear
## 1  AILS -60.78      1960      0    -2.76247100 1.365378  1983
## 2  AILS -60.78      1988      0    -0.97734640 1.365378  1983
## 3  AILS -60.78      1965      0    -2.44369875 1.365378  1983
## 4  AILS -60.78      1970      0    -2.12492650 1.365378  1983
## 5  AILS -60.78      1975      0    -1.80615425 1.365378  1983
## 6  AILS -60.78      2003      0    -0.02102965 1.365378  1983
##   maxyear      Zfit Zlwr  Zupr Zfit_marg Zlwr_marg Zupr_marg
## 1  2019 10359.383 1647 24293 58053.38      0    225310
## 2  2019  5805.399 1547 11399 22681.62      2     83655
## 3  2019  9246.391 1375 21888 46774.22      0    202355
```

## 4	2019	8244.895	1424	17783	44677.31	1	171358
## 5	2019	7463.765	1843	15006	34070.63	2	131095
## 6	2019	4439.312	1381	8268	17841.82	0	60448

```
pop_predict = popy %>%
  dplyr::group_by(season_starting) %>%
  dplyr::summarise(total_pred = sum(Zfit),
                    min_pred = sum(Zlwr),
                    max_pred = sum(Zupr))

pop_predict.p = ggplot(data = pop_predict) +
  geom_line(aes(x = season_starting, y = total_pred),
            lty = 2, linewidth = 1.1)+
  geom_line(aes(x = season_starting, y = min_pred,
                color = "Model predicted count"), lty = 2, size = 0.8)+
  geom_line(aes(x = season_starting, y = max_pred,
                color = "Model predicted count"), lty = 2, size = 0.8)+
  labs(x = "Year", y = "Total population count") +
  theme_bw()+
  scale_y_continuous(label = comma)+
  labs(subtitle = "Total predicted counts across all sites")+
  guides(color=guide_legend(title="Data source"))+
  theme(legend.position = c(0.7, 0.9))

pop_predict.p
```



```
delta.y = 100 * (pop_predict[61,2] - pop_predict[1,2]) / pop_predict[1,2]
delta.y
```

```
## total_pred
## 1 -72.08179
```

6 Predicting population change with entire posterior distribution

Calculate population change over a 30 year period (~ 3 generations according to the original study)

```
# extract posterior draws of fixed effects and random effects
posterior <- as.matrix(mc2$Sol)
```

```
# collect site-level information
site_and_lat <- nestM3 %>%
  as_tibble() %>%
  select(site_id, ZLat) %>%
  distinct()
```

```
site_and_lat
```

```
## # A tibble: 146 x 2
##   site_id ZLat[,1]
##   <chr>      <dbl>
## 1 AILS      1.37
## 2 AITC      0.349
## 3 AITK      1.39
## 4 ALCO     -0.797
## 5 AMPH      1.43
## 6 ANDE      1.38
## 7 ANVS     -1.06
## 8 ARDL      0.470
## 9 ARMS     -1.82
## 10 BAIL      0.000141
## # i 136 more rows
```

```
# map years which to predict to (standardised scale)
# Here, use 1990 as the first year and 2019 as the last year (30 year change)
```

```
year1 = 1990
year2 = 2019
```

```
first_year <- (year1 - mean(df$year)) / sd(df$year)
last_year <- (year2 - mean(df$year)) / sd(df$year)
```

```
# define function to predict with or without random effects
get_predictions <- function(posterior,
                             site_and_lat,
                             first_year,
```

```

        last_year,
        use_random_effects = FALSE) {
# matrices for predictions at each site in year 1 and year 2
# each row is a prediction from a different posterior sample, each column is a site
pred_pop_per_site.first <- matrix(NA, nrow = nrow(posterior), ncol = nrow(site_and_lat))
pred_pop_per_site.last <- matrix(NA, nrow = nrow(posterior), ncol = nrow(site_and_lat))

for (s in 1:nrow(posterior)) {
  theta <- posterior[s,]
  for (j in 1:nrow(site_and_lat)) {
    site_id <- site_and_lat$site_id[j]
    ZLat <- site_and_lat$ZLat[j]

    # predict pop at site j in first year
    lin_pred <- theta["(Intercept)"] +
      theta["Zseason_starting"] * first_year +
      theta["ZLat"] * ZLat +
      theta["Zseason_starting:ZLat"] * first_year * ZLat
    if (use_random_effects) {
      lin_pred <- lin_pred +
        theta[ str_c("(Intercept).site_id.",site_id) ] +
        theta[ str_c("Zseason_starting.site_id.",site_id) ] * first_year
    }
    pred_pop_per_site.first[s,j] <- exp( lin_pred )

    # predict pop at site j in last year
    lin_pred <- theta["(Intercept)"] +
      theta["Zseason_starting"] * last_year +
      theta["ZLat"] * ZLat +
      theta["Zseason_starting:ZLat"] * last_year * ZLat
    if (use_random_effects) {
      lin_pred <- lin_pred +
        theta[ str_c("(Intercept).site_id.",site_id) ] +
        theta[ str_c("Zseason_starting.site_id.",site_id) ] * last_year
    }
    pred_pop_per_site.last[s,j] <- exp( lin_pred )
  }
}

# sum over sites for population level predictions
pred_pop.first <- rowSums(pred_pop_per_site.first)
pred_pop.last <- rowSums(pred_pop_per_site.last)

# percent change from year1 to year2
pred_pop_change <- 100 * ( pred_pop.last - pred_pop.first ) / pred_pop.first

# outputs
predictions <- list(pop_per_site.first = pred_pop_per_site.first,
  pop_per_site.last = pred_pop_per_site.last,
  pop.first = pred_pop.first,
  pop.last = pred_pop.last,
  pop_change = pred_pop_change)

predictions

```

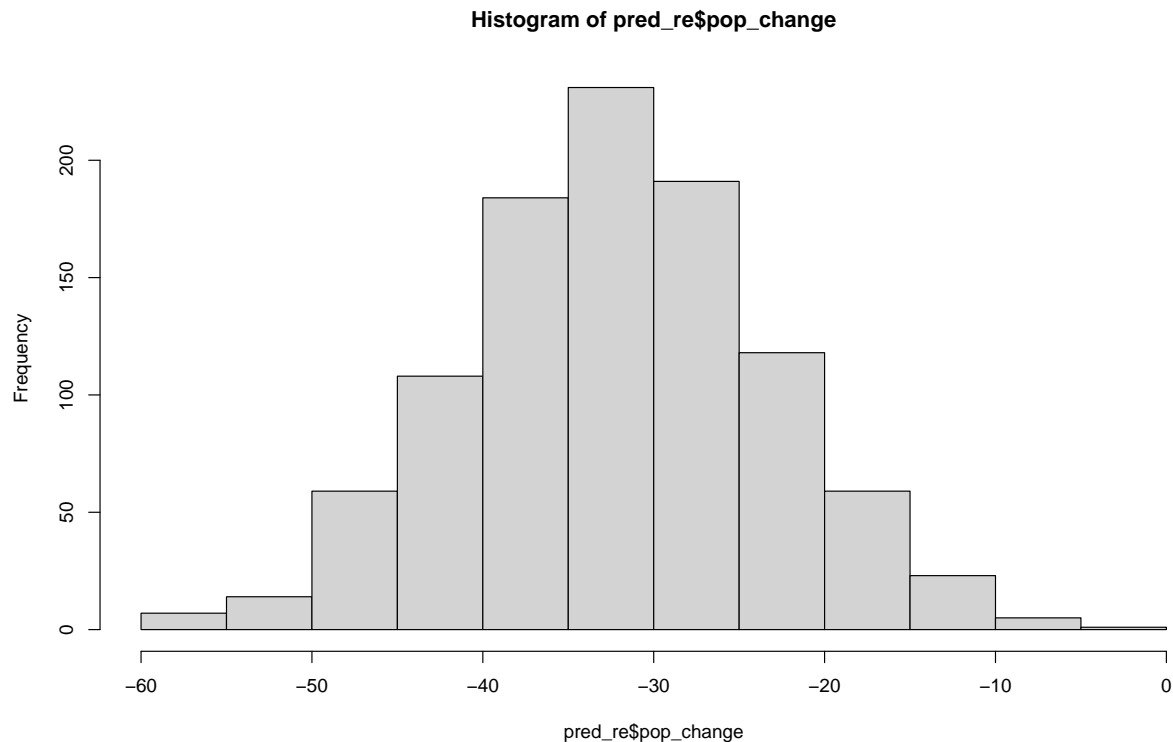
```

}

# Make predictions with and without random effects
pred_re <- get_predictions(posterior, site_and_lat, first_year, last_year,
                           use_random_effects = TRUE)

# Plot histogram of population change using random effects in prediction:
hist(pred_re$pop_change, breaks = 20)

```



```

# can calculate the probability that the population has decreased by
# at least thirty percent with
mean(pred_re$pop_change < -30)

```

```
## [1] 0.603
```

```

# estimated population size in year1 (with random effects)
pred_first = as.data.frame(pred_re$pop.first)
names(pred_first) = "pred_first"
#hist(pred_first$pred_first, breaks = 20)

# estimated population size in year2 (with random effects)
pred_last = as.data.frame(pred_re$pop.last)
names(pred_last) = "pred_last"
#hist(pred_last$pred_last, breaks = 20)

```



```

#-----
# pred_no_re <- get_predictions(posterior, site_and_lat, first_year, last_year,
#                               use_random_effects = FALSE)
#
# # Plot histogram of population change without random effects in prediction:
# hist(pred_no_re$pop_change, breaks = 20)
#
# # estimated population size in year1 (no random effects)
# pred_first_noRE = as.data.frame(pred_no_re$pop.first)
# names(pred_first_noRE) = "pred_first"
#
# # estimated population size in year2 (no random effects)
# pred_last_noRE = as.data.frame(pred_no_re$pop.last)
# names(pred_last_noRE) = "pred_last"
#-----
pop_predict.p +
  geom_point(data = pred_first, aes(x = year1, y = pred_first), col = "red") +
  geom_point(data = pred_last, aes(x = year2, y = pred_last), col = "red")

```

