

Decreasing trends of chinstrap penguin breeding colonies in a region of major and ongoing environmental change: a statistical critique and reanalysis of Krüger (2023)

30-year population change at 91 sites with at least two counts (with accuracy < 5)

Chris Oosthuizen, Murray Christian, Mzabalazo Ngwenya

2023-10-05

Contents

1	Context	1
2	Load packages and set plotting theme	2
3	Load and process MAPPPD data for area 48.1 and 48.2:	2
3.1	Processed data summary	3
3.2	Data distribution figures	3
4	Fit revised GLMM	6
4.1	MCMCglmm diagnostics for mc2	7
5	Predict counts for every year	14
5.1	Conditional model predictions	16
6	Estimate 30-year population change from entire posterior distribution	26
6.1	Plot overall population trend	30

1 Context

This script provides a reanalysis of chinstrap penguin population trends, in the context 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.).

- To reduce extrapolation beyond the range of observed data, this revised analysis restricted predictions of population trends between 1980 and 2019.
- This script evaluated the 30-year population change between 1990 and 2019 at all sites ($n = 91$) with at least two counts (with accuracy < 5) between 1980 and 2019

2 Load packages and set plotting theme

```
# Load packages
library(tidyverse)
library(MCMCglmm)
library(scales)
library(ggforce)
library(colorspace)
library(scales)
library(patchwork)
library(broom.mixed)
```

3 Load and process MAPPPD data for area 48.1 and 48.2:

```
# 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 data to chinstrap penguins only
chins<-subset(df,common_name=="chinstrap penguin")

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

# subset to cammlr_region 48.1 and 48.2
nests <- subset(nests,cammlr_region == "48.1" | cammlr_region == "48.2")

# remove the most uncertain counts (could be very inaccurate - an order of magnitude)
# This is a choice we made for the current analysis.
# Comment this line of code out to run analysis including uncertain counts
nests <- subset(nests, accuracy < 5)

nests = subset(nests, nests$season_starting > 1979)

which(colSums(is.na(nests))>0)
```

```
##   day month
##     7     8
```

```
# some populations had multiple counts over the same season:
# this code summarises the count with the maximum nests
nestM = nests %>%
  group_by(site_id, season_starting) %>%
  slice(which.max(penguin_count)) %>% # take only maximum counts
  dplyr::rename(Lat = latitude_epsg_4326,
                Lon = longitude_epsg_4326,
                nests = penguin_count) %>%
  dplyr::select(season_starting, site_id, nests, Lat, Lon)
```

3.1 Processed data summary

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

summary(as.factor(countsN$ncounts)) # most populations are only counted once
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12   15   16   22   23   24   28   30   31
## 152   35   13    7    4    5    1    4    2    5    2    2    1    1    2    1    1    2    1    1
##   33
##    1
```

```
npops=length(countsN$ncounts[countsN$ncounts>1])
npops # number of populations with more than 2 counts
```

```
## [1] 91
```

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

# Subset to sites with more than 1 count:
nestm3 = subset(nestM2, ncounts>1)
# Subset to sites with counts at least 10 years apart
# nestm3 = subset(nestm3, nestm3$interval > 9)

nestM3 <- nestm3

countspersite = nestM3 %>%
  group_by(site_id) %>%
  summarise(counts = mean(ncounts))

summary(countspersite$counts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   6.725   8.000  33.000
```

3.2 Data distribution figures

```
samplesize = nestM3 %>% group_by(site_id, ncounts) %>% tally()
length(unique(nestM3$site_id))
```

```
## [1] 91
```

```

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

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

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

# 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("Time difference 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

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+'))))) +

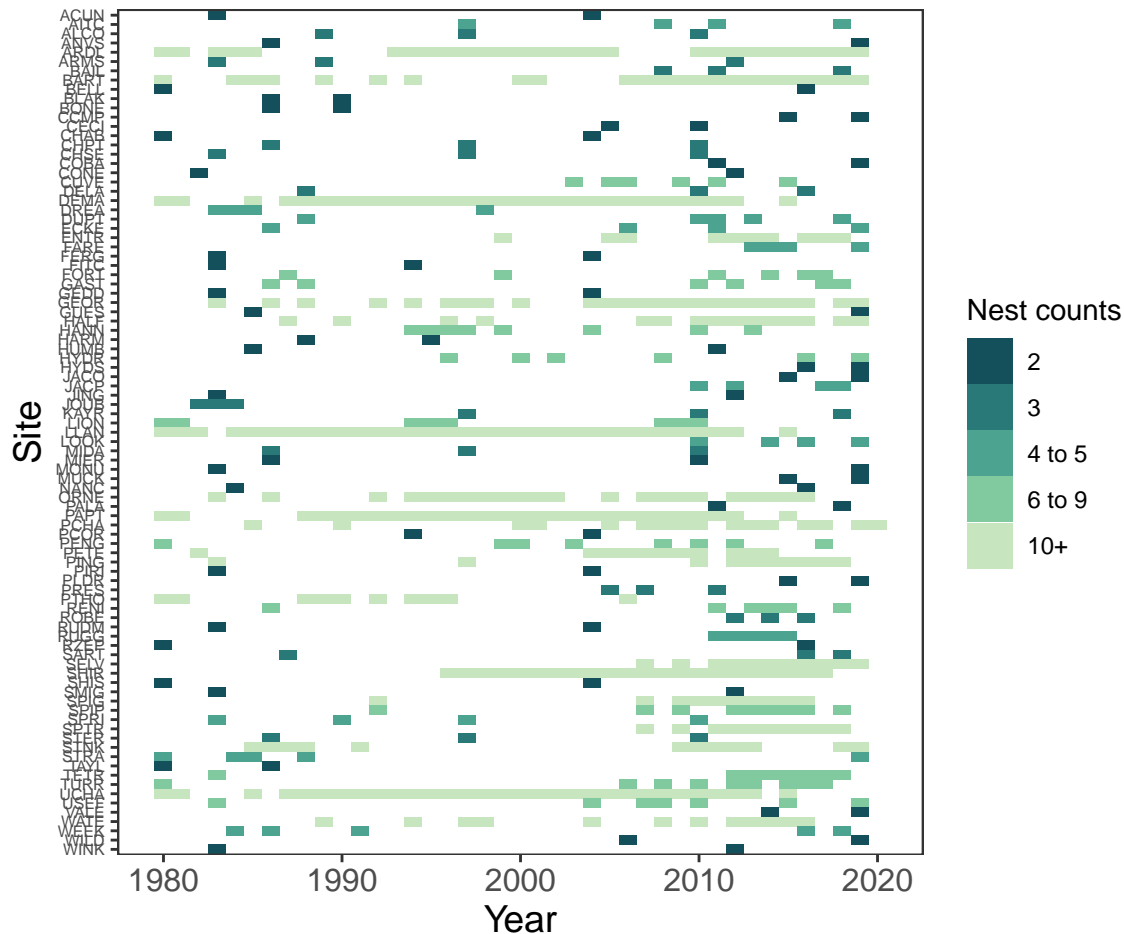
```

```

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

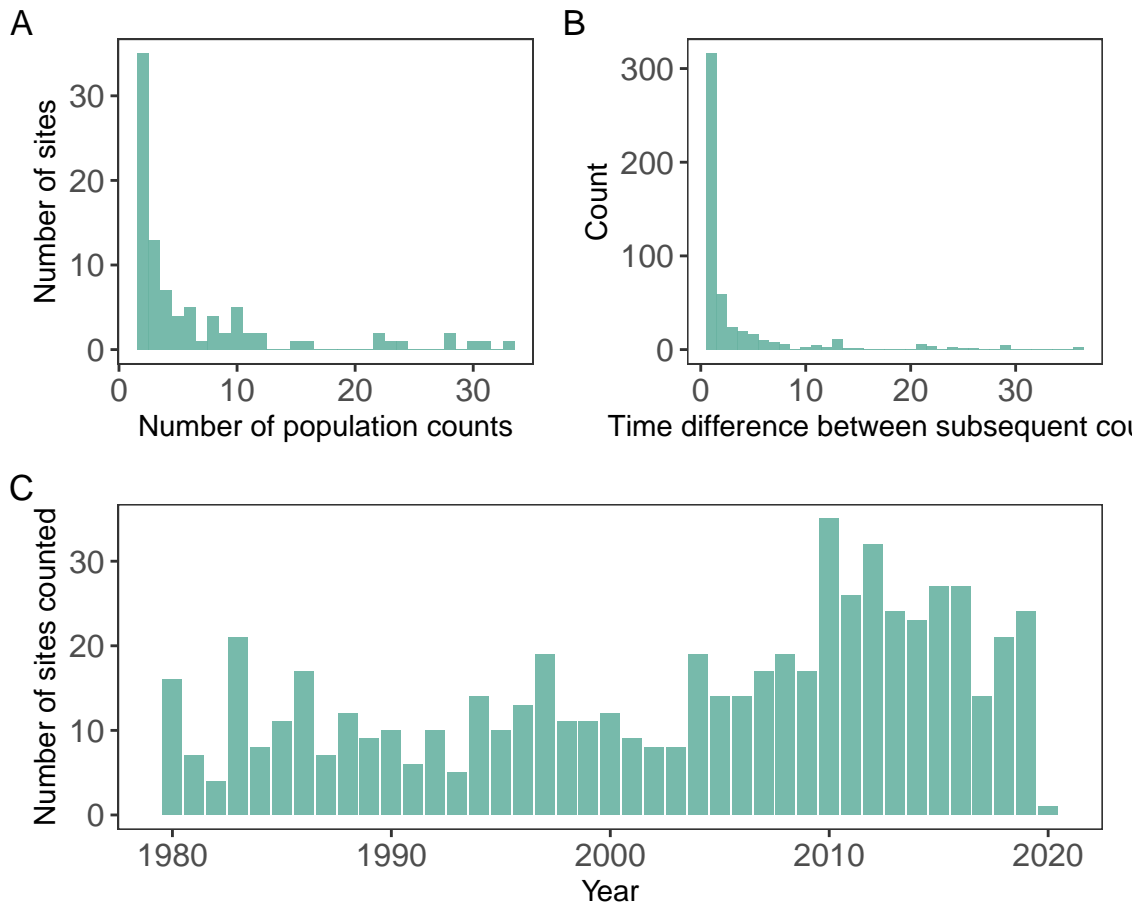


```

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

```

combinedfig



4 Fit revised GLMM

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

```
# Covariates should be standardized.
nestM3$Zseason_starting = scale(nestM3$season_starting)
nestM3$ZLat = scale(nestM3$Lat)

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))))
```

```
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=50000, thin=10, burnin=20000, pr=T,
pl=FALSE, verbose=F, DIC=TRUE, singular.ok=FALSE, saveX=TRUE,
prior=prior, saveZ=TRUE, saveXL=TRUE, slice=FALSE,
ginverse=NULL, trunc=FALSE)
```

```
summary(mc2)
```

```
##
## Iterations = 20001:49991
## Thinning interval = 10
## Sample size = 3000
##
## DIC: 5610.681
##
## G-structure: ~us(1 + Zseason_starting):site_id
##
##               post.mean l-95% CI u-95% CI eff.samp
## (Intercept):(Intercept).site_id      8.2880   5.6874   11.128   533.0
## Zseason_starting:(Intercept).site_id    1.9550   1.0842    2.977   323.2
## (Intercept):Zseason_starting.site_id    1.9550   1.0842    2.977   323.2
## Zseason_starting:Zseason_starting.site_id 0.9873   0.6036    1.444   337.8
##
## R-structure: ~units
##
##      post.mean l-95% CI u-95% CI eff.samp
## units    0.1153  0.09509   0.137   1525
##
## Location effects: nests ~ Zseason_starting * ZLat
##
##               post.mean l-95% CI u-95% CI eff.samp pMCMC
## (Intercept)      5.148480  4.575457  5.760095    3000 <3e-04 ***
## Zseason_starting -0.450026 -0.680357 -0.221427    1445 <3e-04 ***
## ZLat            1.561350  1.063017  2.105240    2518 <3e-04 ***
## Zseason_starting:ZLat -0.187548 -0.391298 -0.001564    1431 0.0627 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.1 MCMCglmm diagnostics for mc2

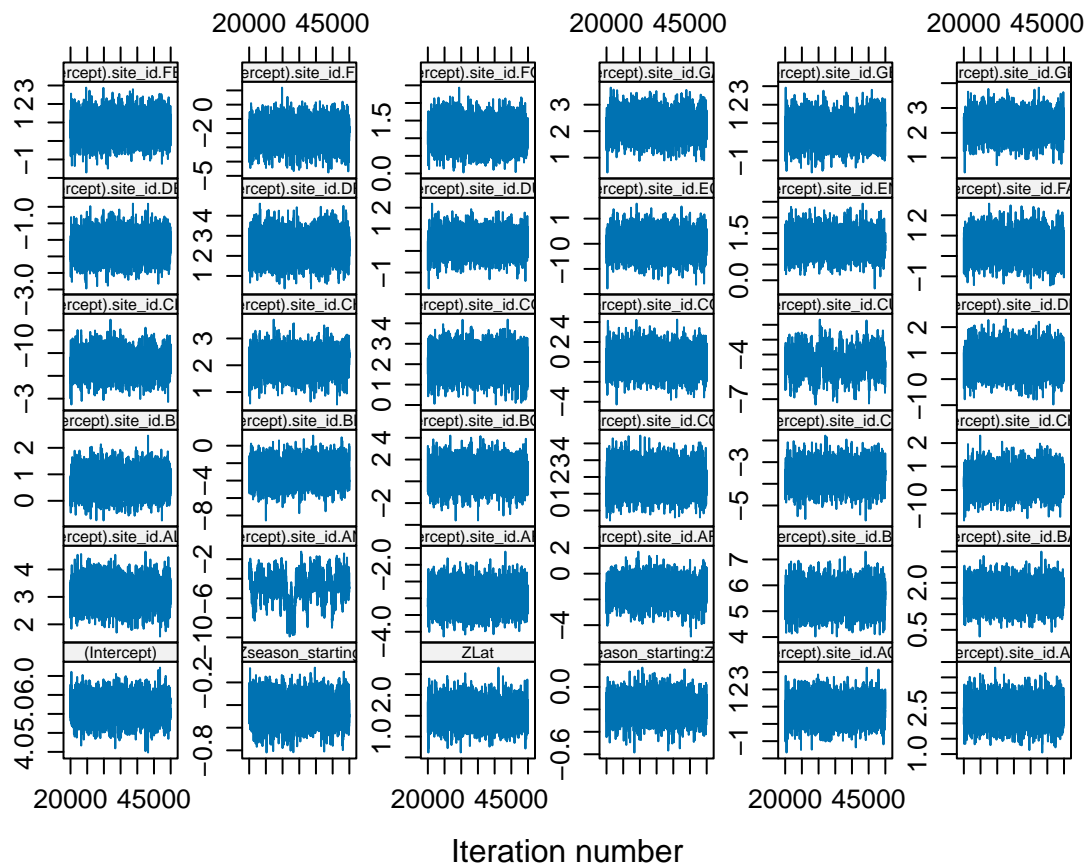
```
# Assessing model convergence. We do this separately for both fixed
# and random effects. The trace plot should look like a fuzzy caterpillar

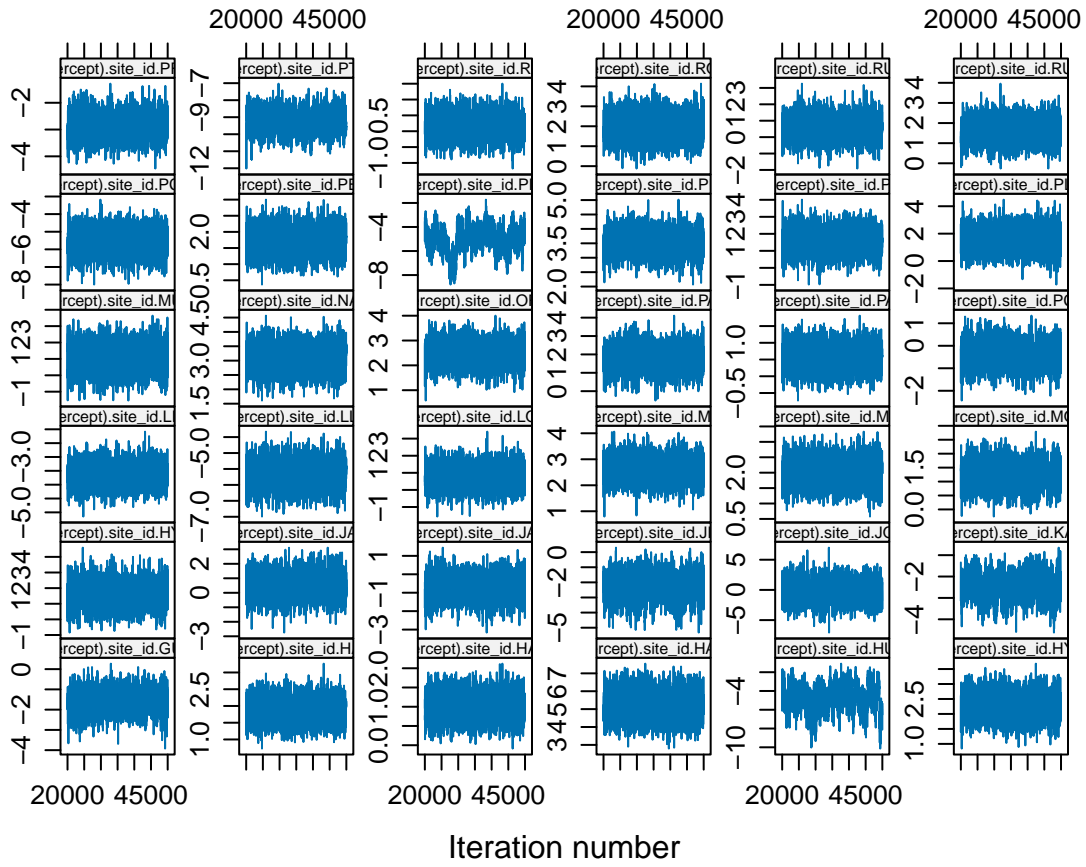
# effective sample size
coda::effectiveSize(mc2$VCV)
```

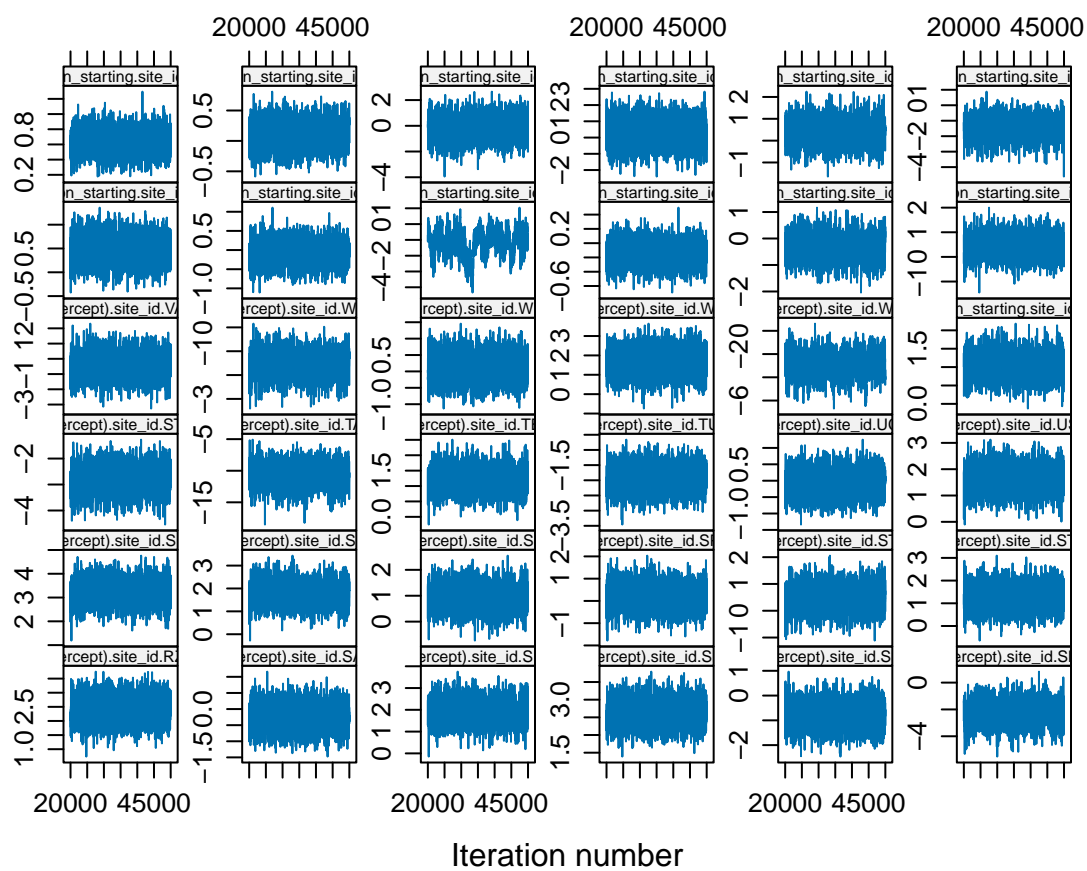
```
##               (Intercept):(Intercept).site_id
##                               533.0215
##      Zseason_starting:(Intercept).site_id
##                               323.2184
```

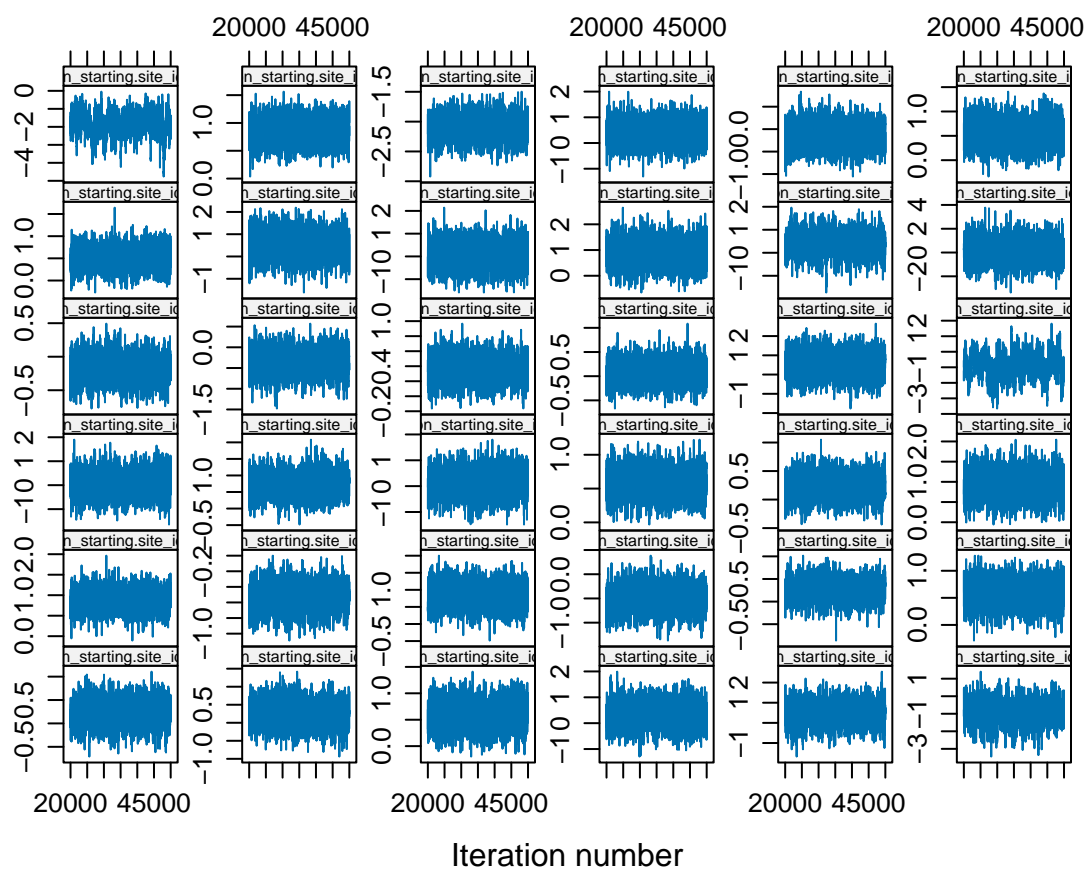
```
##      (Intercept):Zseason_starting.site_id
##                               323.2184
## Zseason_starting:Zseason_starting.site_id
##                               337.8122
##                               units
##                               1525.0693
```

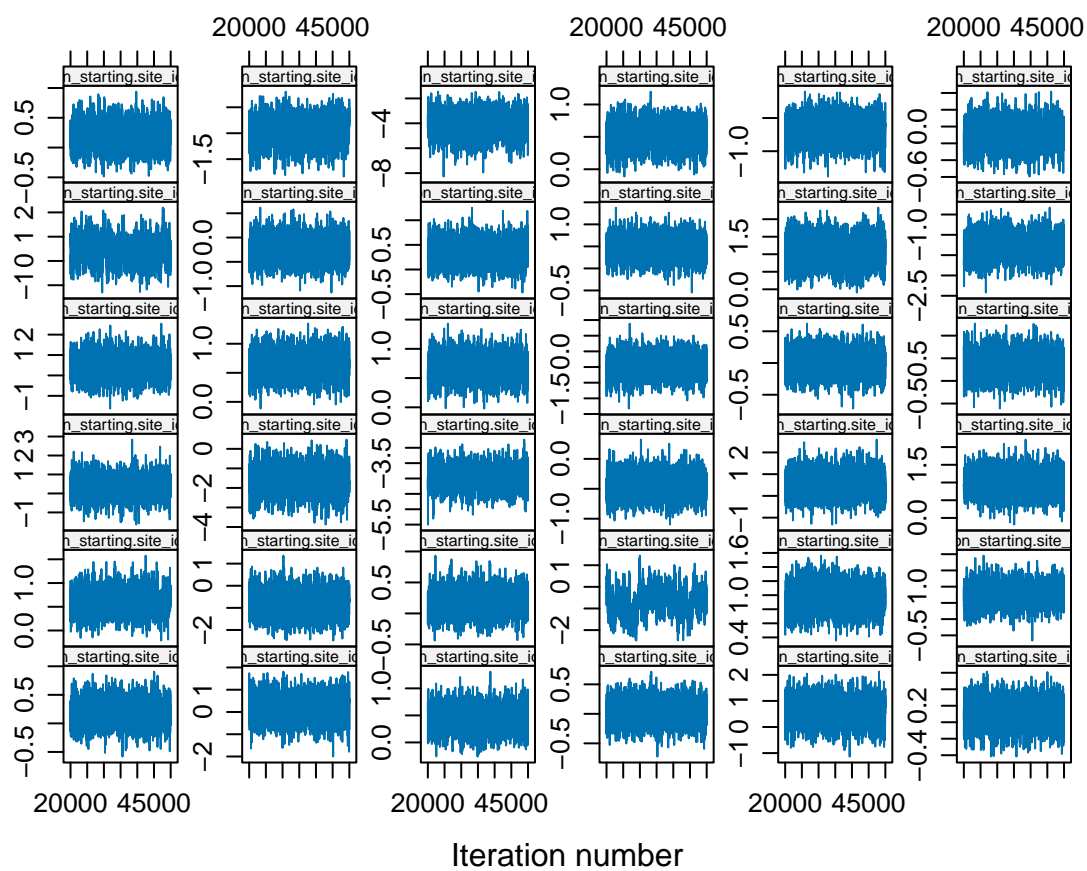
```
# check that the mcmc chain is mixing well - should be "white noise"
lattice::xyplot(as.mcmc(mc2$Sol), layout=c(6,6), par.strip.text=list(cex=0.5))
```

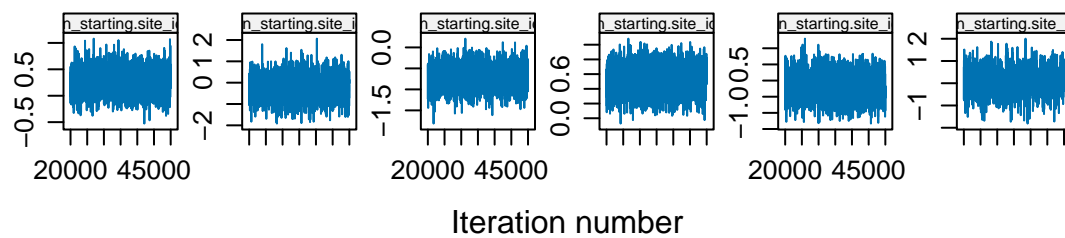




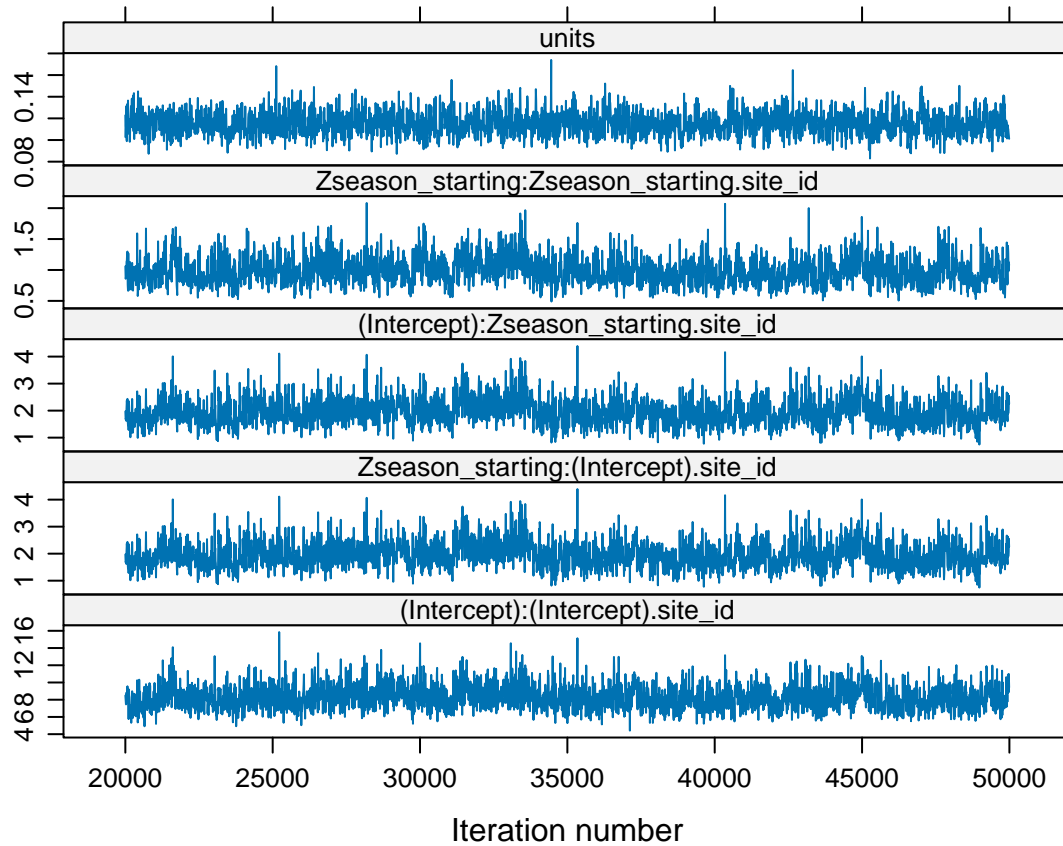








```
# the variance components
lattice::xyplot(as.mcmc(mc2$VCV), par.strip.text=list(cex=0.8))
```



```
# from MCMC Course notes (page 60):
diag(autocorr(mc2$VCV)[2, , ]) # low autocorrelation
```

```
##           (Intercept):(Intercept).site_id
##                                0.1189048
##      Zseason_starting:(Intercept).site_id
##                                0.2208492
##      (Intercept):Zseason_starting.site_id
##                                0.2208492
## Zseason_starting:Zseason_starting.site_id
##                                0.2605555
##                                units
##                                0.2095699
```

5 Predict counts for every year

```
# construct 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
##      2003.25817      -63.15355
```

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

```
## season_starting      Lat
##      11.628821      1.291479
```

```
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(1980:2019)) # 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  ACUN -60.761      1980      0      -2.000045  1.8525633
## 2  AITC -62.407      1980      0      -2.000045  0.5780551
## 3  ALCO -64.240      1980      0      -2.000045 -0.8412484
## 4  ANVS -64.661      1980      0      -2.000045 -1.1672314
## 5  ARDL -62.213      1980      0      -2.000045  0.7282705
## 6  ARMS -65.884      1980      0      -2.000045 -2.1142080
```

```
# 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: 91 x 3
##   site_id minyear maxyear
##   <chr>    <int>   <int>
## 1 ARDL     1980    2019
## 2 BART     1980    2019
## 3 BELL     1980    2016
```

```
## 4 CHAB      1980    2004
## 5 DEMA      1980    2015
## 6 LION      1980    2010
## 7 LLAN      1980    2015
## 8 PAPT      1980    2015
## 9 PENG      1980    2017
## 10 PTHO     1980    2006
## # i 81 more rows
```

```
poppy = merge(poppy, first_last_season)

length(unique(poppy$site_id))
```

```
## [1] 91
```

```
popypred <- data.frame(predict(mc2,
                             newdata=poppy,
                             type="response",
                             marginal=NULL,      # crucial, and not default code.
                             # marginal=~us(1 + Zseason_starting):site_id,
                             interval="prediction",
                             posterior="all"))

head(popypred)
```

```
##      fit   lwr   upr
## 1 4728.206 1099  9828
## 2 5096.327 1532 10310
## 3 5455.859 1786 10579
## 4 5887.719 2095 10997
## 5 6283.527 2123 11570
## 6 12067.976 1054 30207
```

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

5.1 Conditional model predictions

```
# How accurate are the predictions relative to observed data?
required_n_pages = round(80/16)+1
for(i in 1:required_n_pages){

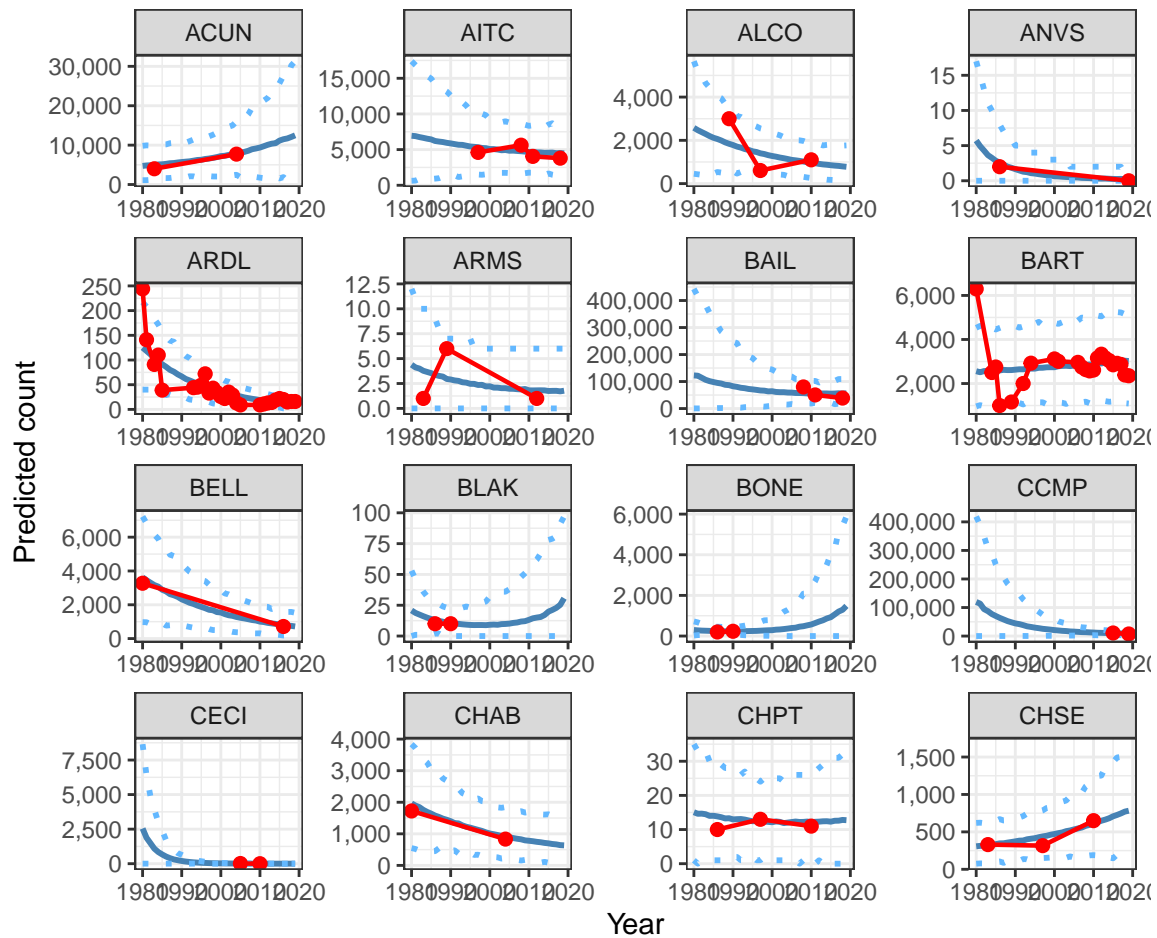
print(ggplot(data = poppy) +
      geom_line(aes(x = season_starting, y = Zfit),
                col = "steelblue", linewidth=1.04) +
      geom_line(aes(x = season_starting, y = Zlwr),
                col = "steelblue1", linetype="dotted", linewidth = 1.02) +
      geom_line(aes(x = season_starting, y = Zupr),
                col = "steelblue1", linetype="dotted", linewidth=1.02) +
```

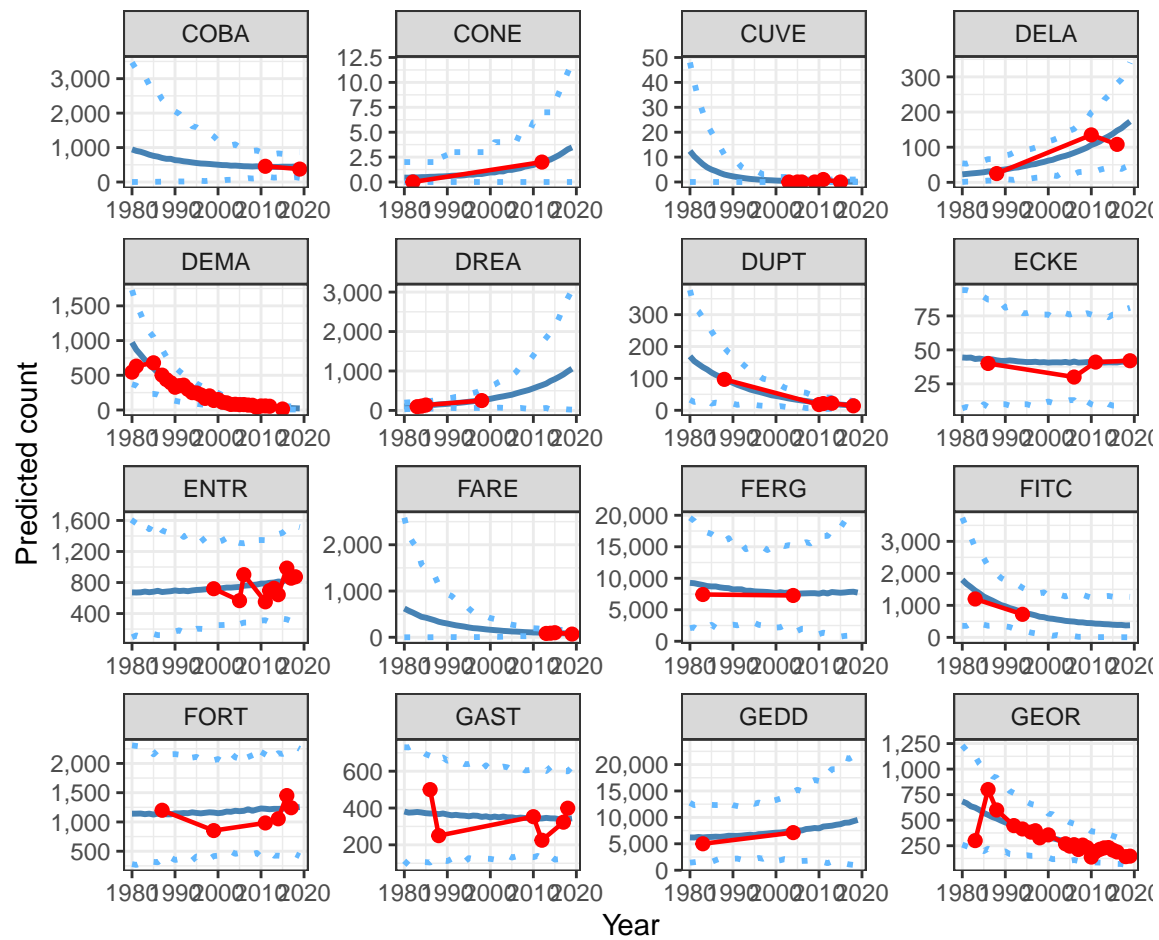


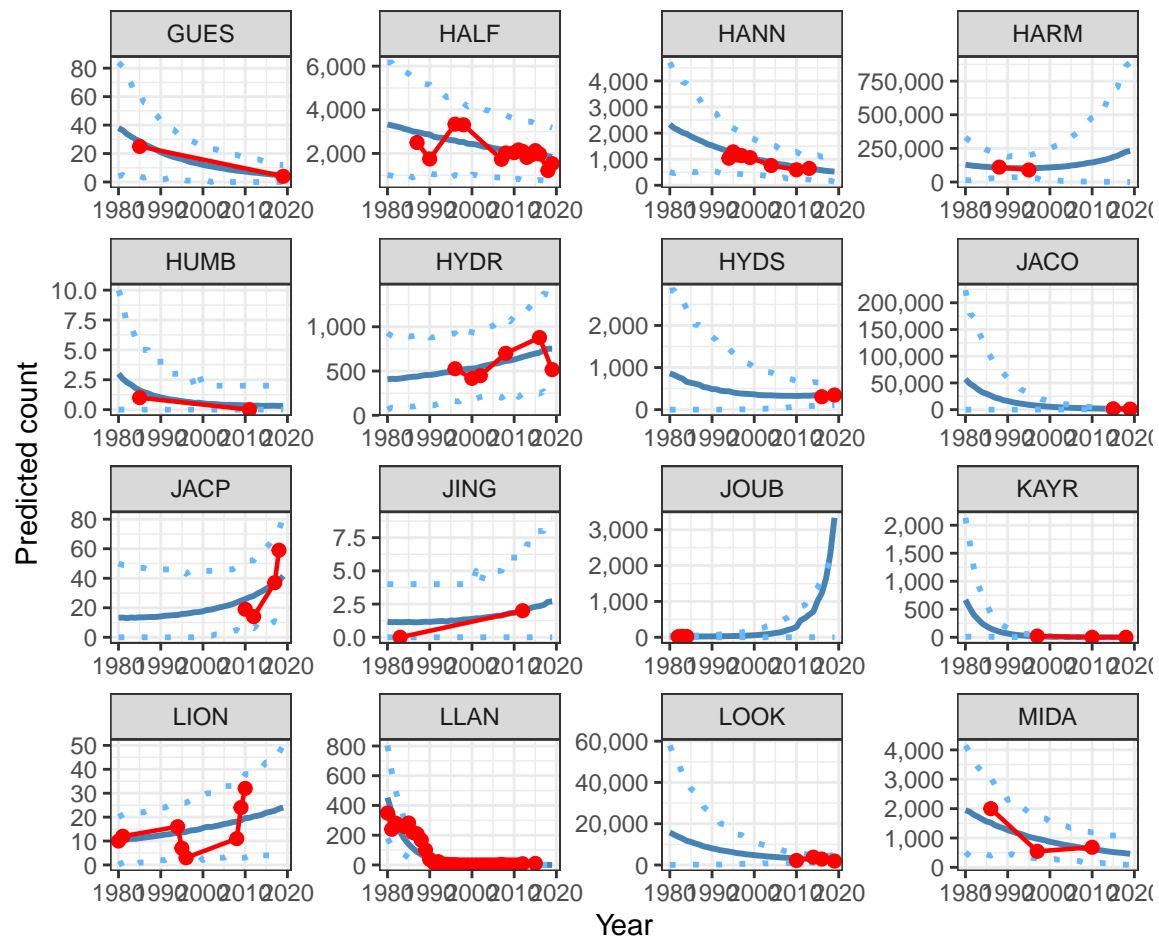
```

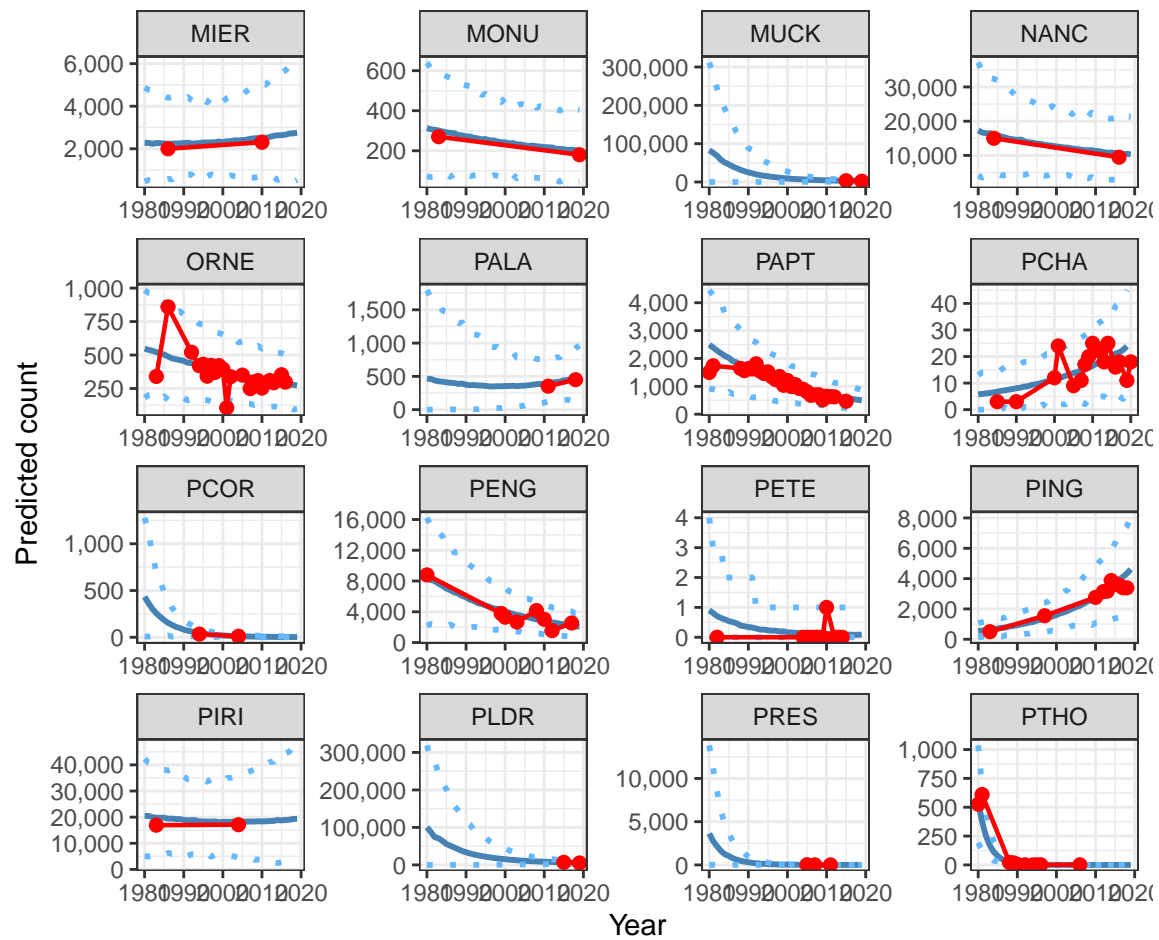
geom_point(data = nestm3, aes(season_starting, y = nests),
           color = "red", cex = 2) +
geom_line(data = nestm3, aes(season_starting, y = nests),
          color = "red", linewidth=0.8) +
theme_bw() +
xlab("Year") +
ylab("Predicted count") +
scale_y_continuous(label = comma)+
# theme(strip.text = element_text(size = 1.5)) +
facet_wrap_paginate(~ site_id, ncol = 4, nrow = 4,
                    page = i,
                    scales = 'free'))}

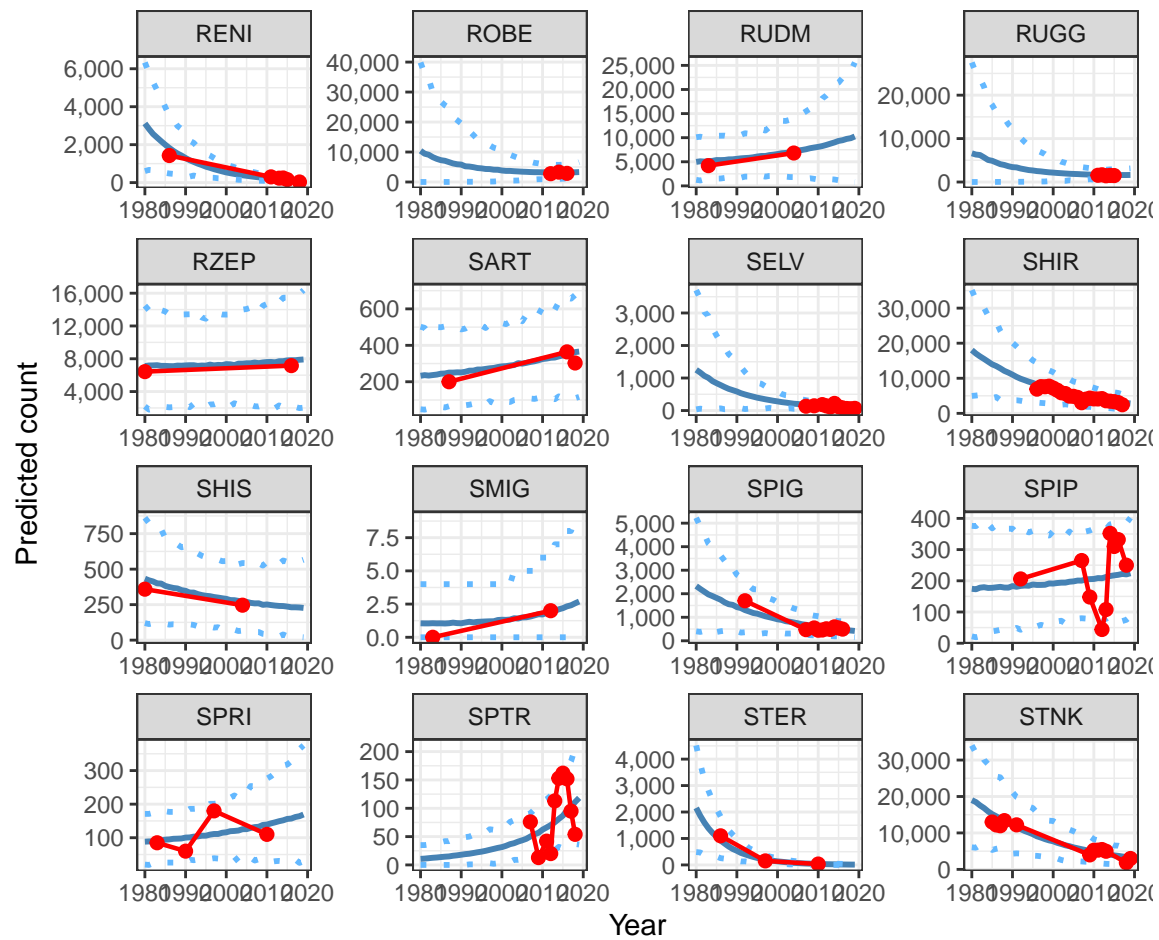
```

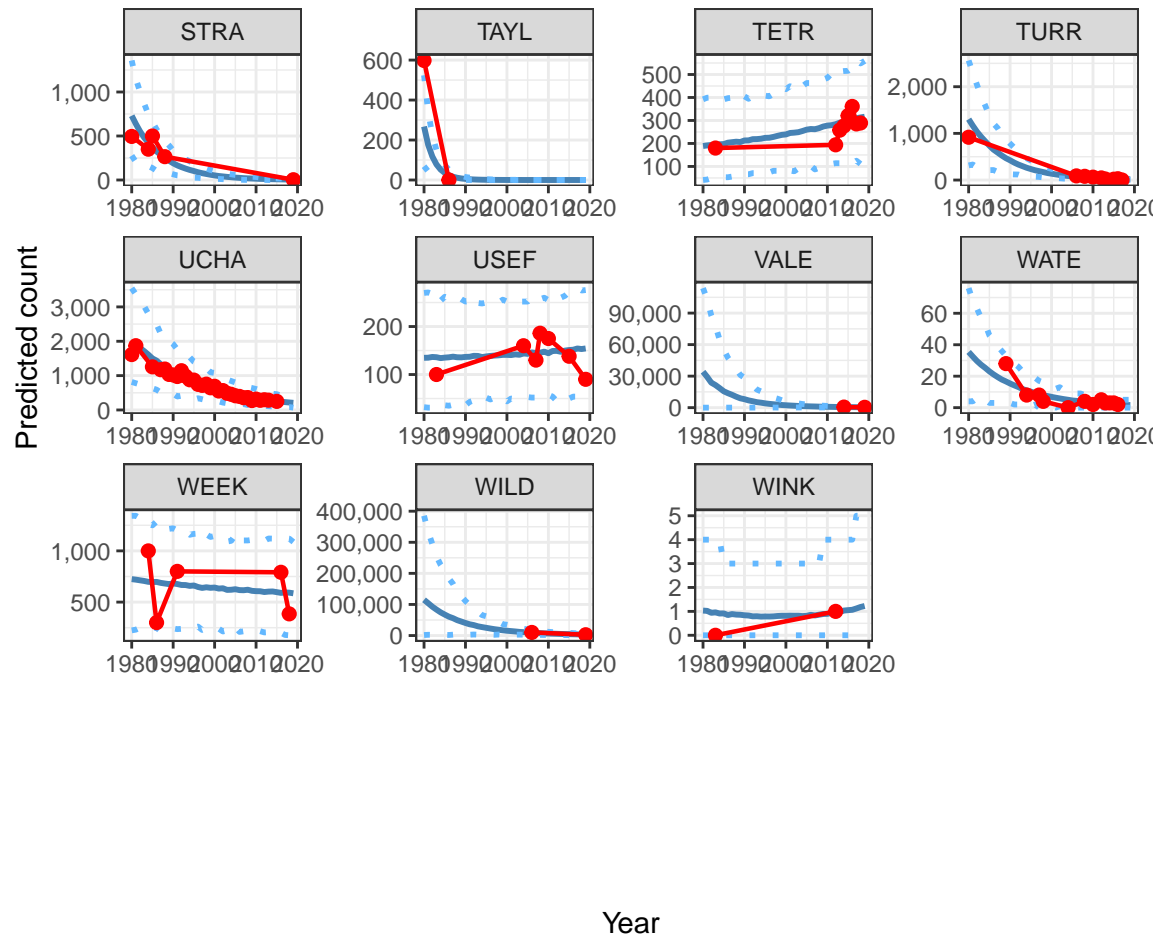












```
# Predictions are generally good
```

```
# Look at some sites with temporally sparse data and strong extrapolation
nestM3 %>% dplyr::filter(site_id == "MUCK")
```

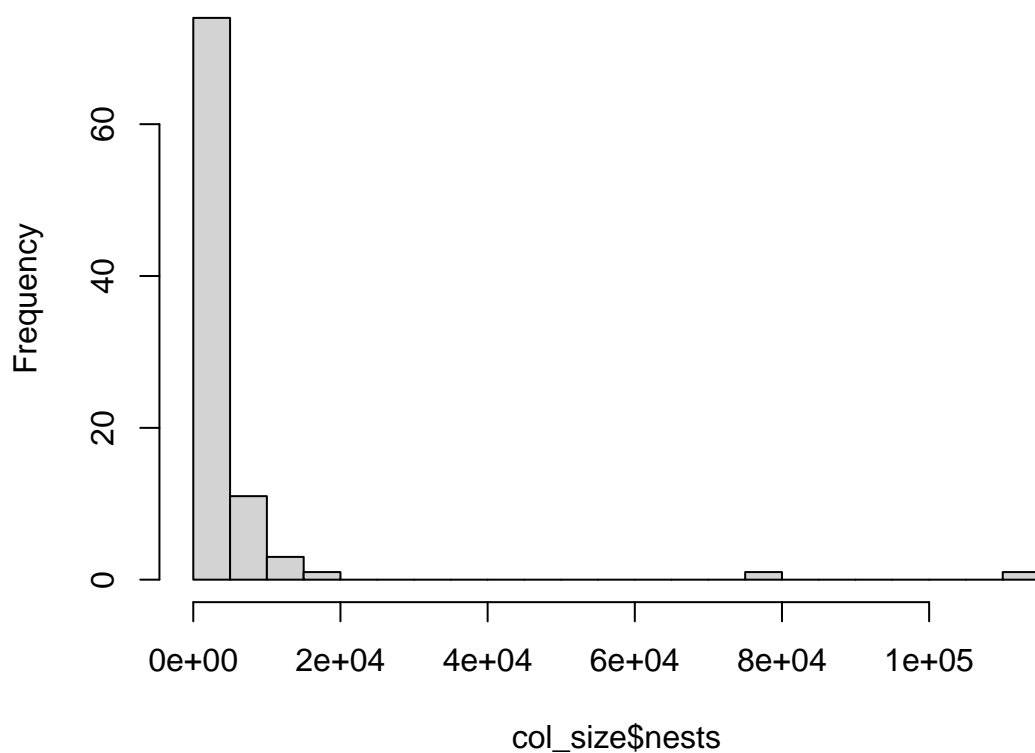
```
##   site_id   Lat   Lon season_starting nests ncounts minseason maxseason
## 1   MUCK -61.157 -54.86          2015  3438         2      2015      2019
## 2   MUCK -61.157 -54.86          2019  2114         2      2015      2019
##   interval Zseason_starting      ZLat
## 1         4          1.009718 1.545938
## 2         4          1.353691 1.545938
```

```
# some unreasonably high values:
```

```
col_size = nestM3 %>%
  group_by(site_id) %>%
  slice(which.max(nests))
```

```
hist(col_size$nests, breaks = 30)
```

Histogram of col_size\$ nests



```
# extract random effects from MCMCglmm
# https://stackoverflow.com/questions/64562052/extract-random-effects-from-mcmcglmm
```

```
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 ACUN          0.798          1.05          0.650
## 2 AITC          2.37          0.470         0.394
## 3 ALCO          3.15         -0.0677        0.448
## 4 ANVS         -4.59         -1.07          1.51
## 5 ARDL         -2.98         -0.142         0.369
```

```
## 6 ARMS -1.33 -0.320 0.867
## # 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 (-)

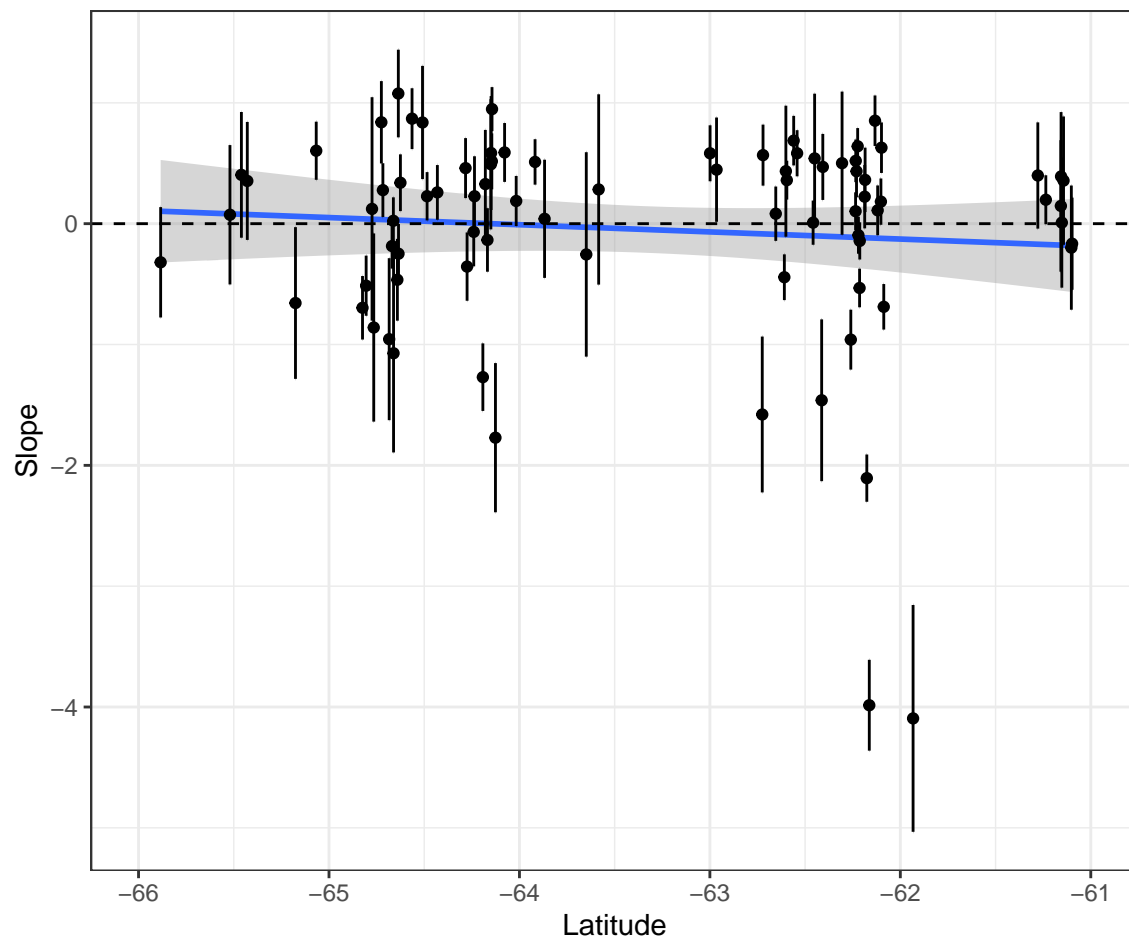
names(re) = c("site_id", "est_int", "estZss",
              "se_int", "seZss")
# add latitude
nestM3_lat = dplyr::select(nestM3, Lat, site_id) %>%
  dplyr::distinct(site_id, Lat)

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

# plot relationship between slope and latitude

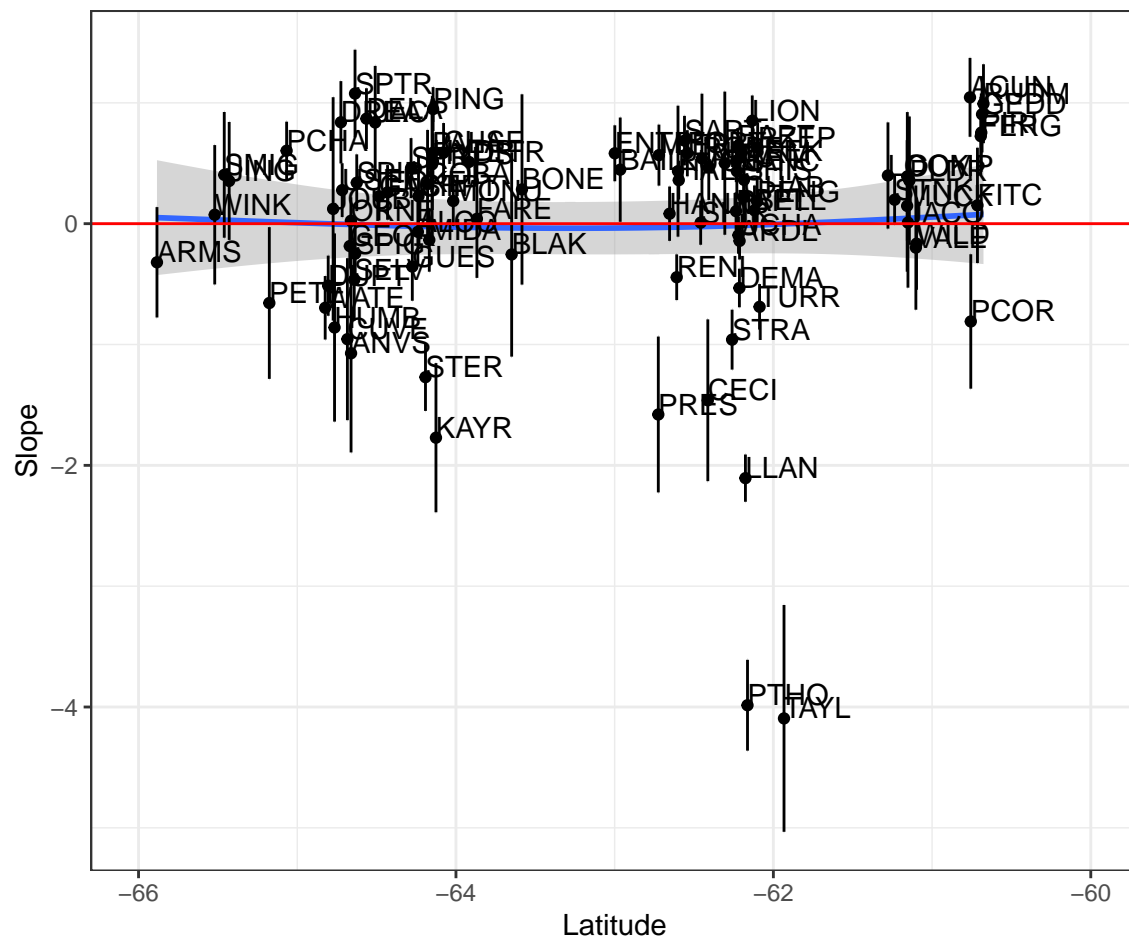
lat_plot = ggplot(data = re, aes(x = Lat, y = estZss))+
  stat_smooth(method="gam", formula=y~s(x,k=2))+
  # geom_smooth(method='lm', formula= y~x)+
  geom_point()+
  geom_errorbar(aes(ymin=estZss-seZss,
                   ymax=estZss+seZss))+
  theme_bw()+
  ylab("Slope")+
  scale_x_continuous(lim = c(-66,-61), breaks = seq(-66, -61, by = 1))+
  xlab("Latitude")+
  geom_hline(yintercept=0,
            color = "black", lty = 2)

lat_plot
```

```
## Save Plot
# pdf("./figures/MS_Latitude.pdf",
#      useDingbats = FALSE, width = 5, height = 4)
# plot(lat_plot)
# dev.off()

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



6 Estimate 30-year population change from entire posterior distribution

```
# 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: 91 x 2
##   site_id ZLat[,1]
##   <chr>      <dbl>
## 1 ACUN      1.85
## 2 AITC      0.578
```

```
## 3 ALCO      -0.841
## 4 ANVS      -1.17
## 5 ARDL       0.728
## 6 ARMS      -2.11
## 7 BAIL       0.146
## 8 BART       0.720
## 9 BELL       0.801
## 10 BLAK     -0.384
## # i 81 more rows
```

```
# map years which to predict to (standardised scale)
# Here, the first year is 1990 and the last year is 2020 (30 year change)

year1 = 1990
year2 = 2019

first_year <- (year1 - mean(nestM3$season_starting)) / sd(nestM3$season_starting)
last_year <- (year2 - mean(nestM3$season_starting)) / sd(nestM3$season_starting)

# 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 (1990) and year 2 (2020)
  # 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
}

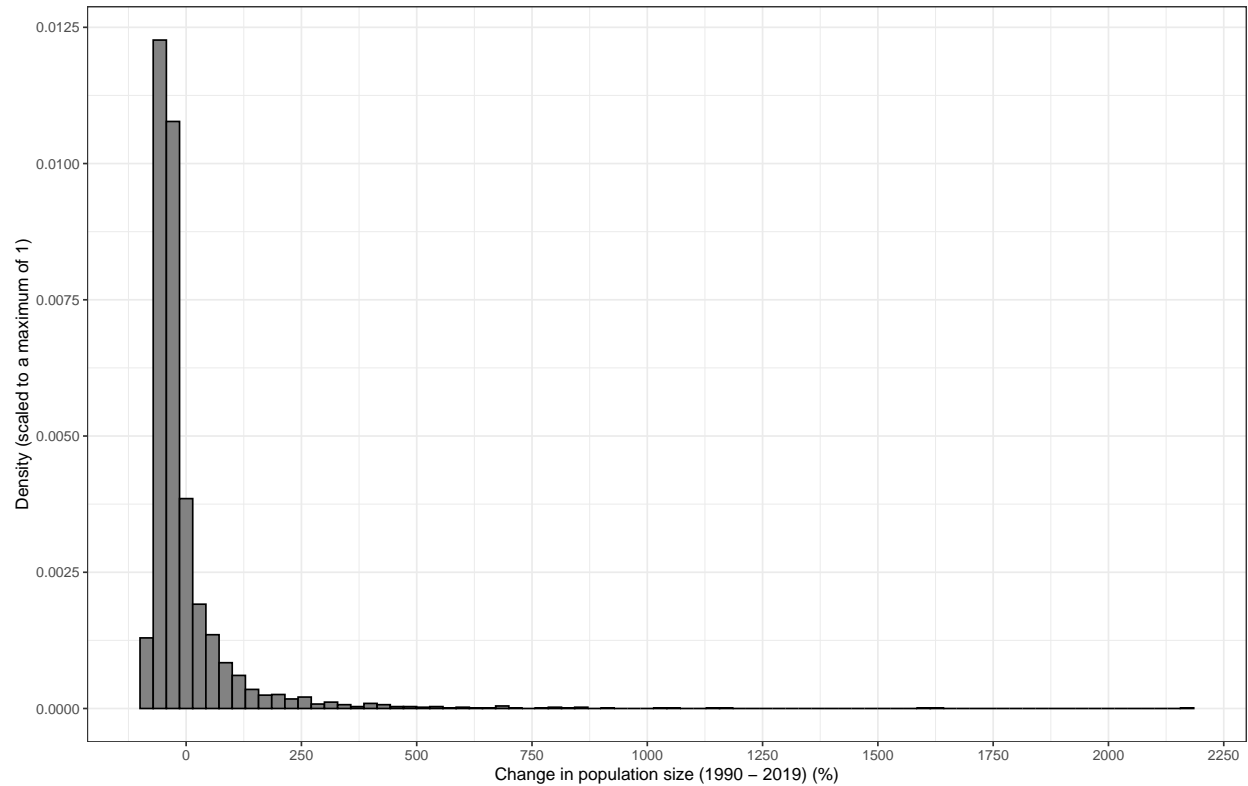
# Make predictions with and without random effects
pred_re <- get_predictions(posterior, site_and_lat, first_year, last_year,
                          use_random_effects = TRUE)

# Plot histogram of population change using random effects in prediction:

hist_growth =
  ggplot(data = data.frame(pred_re$pop_change), aes(x = pred_re.pop_change,
                                                    after_stat(density))) +
  geom_histogram(bins = 80, colour = "black", fill = "grey51") +
  theme_bw() +
  scale_x_continuous(n.breaks = 10) +
  labs(y= "Density (scaled to a maximum of 1)",
       x = "Change in population size (1990 - 2019) (%)")

hist_growth

```



```
# can calculate the probability that the population has decreased by
# at least thirty percent with
sum(pred_re$pop_change < -30)/2000 # McElreath Chapter 3
```

```
## [1] 0.8495
```

```
mean(pred_re$pop_change < -30)
```

```
## [1] 0.5663333
```

```
mean(pred_re$pop_change < 0)
```

```
## [1] 0.761
```

```
quantile(pred_re$pop_change,c(0.05,0.95))
```

```
##          5%          95%
## -69.26077 162.44233
```

```
length(unique(nestm3$site_id))
```

```
## [1] 91
```

```

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

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

```

6.1 Plot overall population trend

```

# How many penguins were there, per year, across all sites?
# We don't know, as we only have intermittent counts.

pop_predict = popy %>%
#   dplyr::filter(site_id != "HARM") %>%
#   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 = 1, linewidth = 1.1)+
  geom_line(aes(x = season_starting, y = min_pred), lty = 2, linewidth = 0.8)+
  geom_line(aes(x = season_starting, y = max_pred), lty = 2, linewidth = 0.8)+
  labs(x = "Year", y = "Total population count") +
  theme_bw()+
  scale_y_continuous(label = comma)

#pop_predict.p

pop_predict.p = pop_predict.p +
  geom_jitter(data = pred_first, aes(x = year1, y = pred_first),
             col = "black", size = 1, height = 0, width = 0.5,
             alpha = 0.05, stroke = NA) +
  geom_jitter(data = pred_last, aes(x = year2, y = pred_last),
             col = "black", size = 1, height = 0, width = 0.5,
             alpha = 0.05, stroke = NA) +
  scale_y_continuous(lim = c(0, 3000000), labels = scales::comma)

```

```

## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.

```

```
pop_predict.p
```

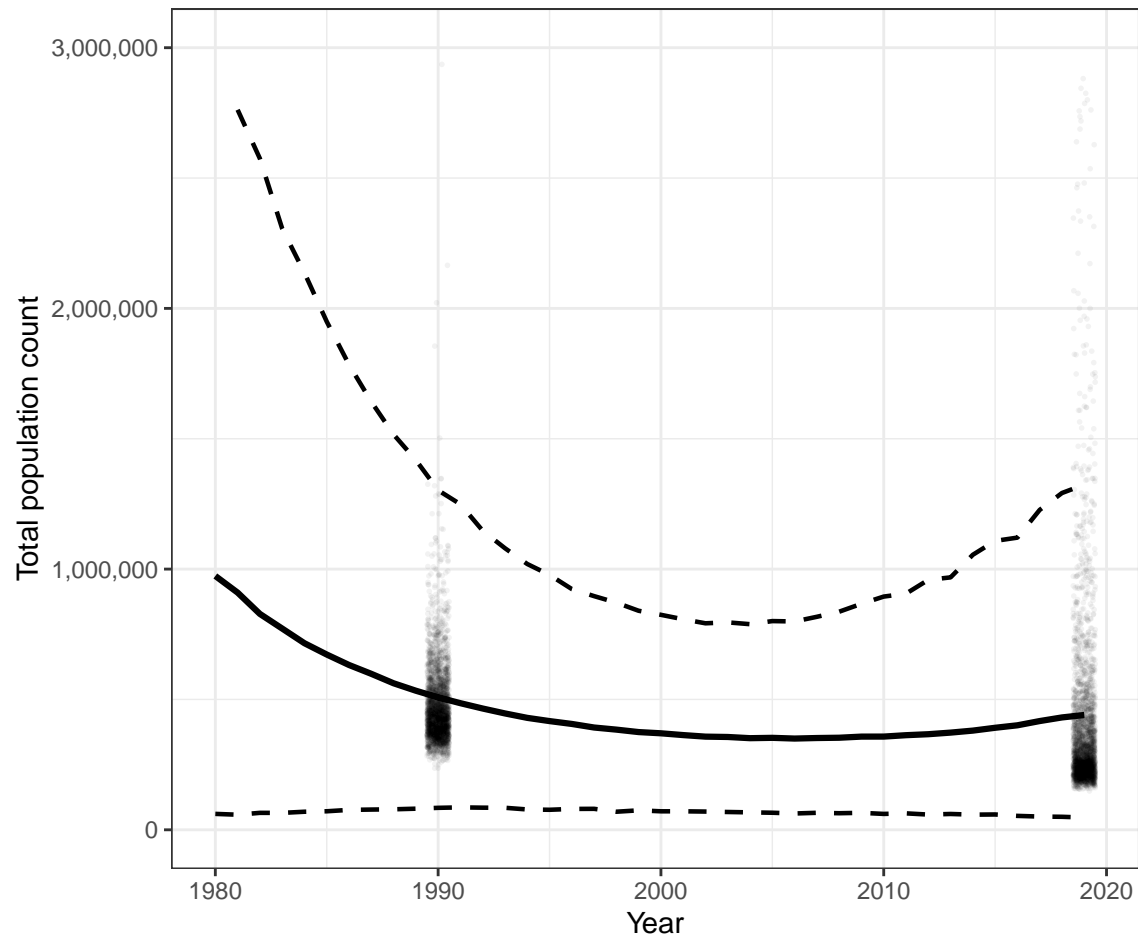
```

## Warning: Removed 1 row containing missing values ('geom_line()').

## Warning: Removed 1 rows containing missing values ('geom_point()').

## Warning: Removed 19 rows containing missing values ('geom_point()').

```



```
## Save Plot
# pdf("./figures/MS_Population_change.pdf",
#      useDingbats = FALSE, width = 10, height = 4)
# cowplot::plot_grid(pop_predict.p, hist_growth, labels = c('A', 'B'), ncol = 2, label_size = 12)
# dev.off()
```