**Gentoo Penguin Abundance Model - Core Pipeline**

**Overview**

This script(s) implements a Bayesian hierarchical model to estimate Gentoo penguin (*Pygoscelis papua*) abundance across sites in the Western Antarctic Peninsula from 1970-2023. The model handles irregular count data, varying observation accuracy, and missing observations to produce robust abundance estimates with uncertainty quantification.

**Core Pipeline Components**

**1. Data Processing (gentoo\_presence\_absence\_assumptions.R)**

**2. JAGS Model Specification (Mapppd-Gentoo-Abundance-Model-JAGS\_Commented.R)**

**3. Model Fitting & Parameter Extraction (JAGS-Gentoo-Abundance-Model.Rmd)**

**1. Data Processing: Generate Presence/Absence Assumptions**

**MAPPPD Data Source**

The analysis uses the **Mapping Application for Penguin Populations and Projected Dynamics (MAPPPD)** database, accessed via the mapppdr R package:

r

library(mapppdr)

*# Core data tables*

mapppdr::penguin\_obs *# Count observations with accuracy ratings*

mapppdr::sites *# Site locations and metadata*

**Data Processing Workflow**

**Step 1: Extract Gentoo Observations**

r

min\_season <- 1970

max\_season <- 2023

species <- "GEPE" *# Gentoo penguin species code*

*# Filter observations*

SiteList <- mapppdr::penguin\_obs %>%

filter(count > 0 & species\_id == species &

season >= min\_season & season <= max\_season) %>%

mutate(season\_relative = season - min\_season + 1) %>%

group\_by(site\_id) %>%

summarise(initial\_season = min(season\_relative))

**Step 2: Create Site × Season Matrix**

r

*# Template for all possible site-season combinations*

w\_template <- SiteList %>%

uncount(n\_seasons) %>% *# Expand each site by number of seasons*

mutate(season\_relative = rep(1:n\_seasons, n\_sites))

**Step 3: Apply Presence/Absence Assumptions**

**Key Assumptions Made:**

* **Gap Filling**: Missing data between observed breeding events filled based on adjacent observations
* **Fill Rules**:
  + Between (present, present) → assume present
  + Between (absent, present) → assume absent
  + Between (present, absent) → assume present
  + Leading/trailing gaps → extend nearest observation

r

w\_df <- penguin\_obs %>%

group\_by(site\_id) %>%

tidyr::fill(w, .direction = "downup") %>% *# Apply gap-filling rules*

ungroup()

*# Convert to matrix for JAGS model*

w <- w\_df %>%

pivot\_wider(names\_from = season\_relative, values\_from = w) %>%

as.matrix()

**Step 4: Process Count Data**

**Count Type Handling:**

* **Adults**: Converted to nest estimates (accuracy +3, max accuracy = 5)
* **Nests**: Used directly
* **Chicks**: Used with breeding success model

**Accuracy to Precision Conversion:**

r

*# Predefined accuracy-to-precision mapping*

mutate(precision = case\_when(

accuracy == 1 ~ 1/0.02490061^2, *# Highest precision*

accuracy == 2 ~ 1/0.04955838^2,

accuracy == 3 ~ 1/0.1201131^2,

accuracy == 4 ~ 1/0.2212992^2,

accuracy == 5 ~ 1/0.4472728^2 *# Lowest precision*

))

**Data Partitioning:**

* **Initial Abundance** (abundance\_initial): First observed count at each site
* **Nest Counts** (abundance\_nests): Subsequent nest observations
* **Chick Counts** (abundance\_chicks): Chick observations for breeding success

**2. JAGS Model Specification**

**Model Structure**

The JAGS model (jags\_model.jags) implements a hierarchical Bayesian framework with multiple linked components:

**Core State Variable**

lz[i,t] = log(abundance) at site i, season t

**Prior Specifications**

# Process variance

sigma ~ dunif(0, 1)

tau <- pow(sigma, -2)

# Growth rate intercept

beta ~ dunif(-0.5, 0.5)

# Site-specific random effects

eta[i] ~ dnorm(0, tau\_site)

sigma\_site ~ dunif(0, 1)

# Seasonal random effects

epsilon[t] ~ dnorm(0, tau\_season)

sigma\_season ~ dunif(0, 1)

# Initial abundance (vague prior)

lz[i, s[i]] ~ dnorm(0, 0.001)

**Observation Models**

**1. Initial Season Observations:**

y\_i[i] ~ dnorm(mu\_y\_i[i], precision\_i[i])

mu\_y\_i[i] <- lz[i, s[i]] - 1/(2 \* precision\_i[i])

**2. Nest Count Observations:**

y\_n[i] ~ dnorm(mu\_y\_n[i], precision\_n[i])

mu\_y\_n[i] <- lz[site\_n[i], season\_n[i]] - 1/(2 \* precision\_n[i])

**3. Chick Count Observations (with breeding success):**

N[i] <- 2 \* round(exp(lz[site\_c[i], season\_c[i]])) # Expected pairs

z\_c[i] ~ dbin(alpha[i], N[i]) # Successful chicks

y\_c[i] ~ dnorm(mu\_y\_c[i], precision\_c[i]) # Observed chick count

# Breeding success prior

alpha[i] ~ dbeta(a, b) # where a,b parameterized for mean=0.5, sd=0.25

**Process Model (Population Dynamics)**

# Growth rate combining fixed and random effects

zr[i,t] <- beta + eta[i] + epsilon[t]

# Population dynamics (forward in time)

for t in (s[i] + 1):n\_seasons:

lz[i,t] ~ dnorm(mu\_lz[i,t], tau)

mu\_lz[i,t] <- lz[i, t-1] + zr[i,t] - 1/(2 \* tau)

# Population dynamics (backward in time for pre-initial seasons)

for t in 1:(s[i] - 1):

lz[i, s[i] - t] ~ dnorm(mu\_lz[i, s[i] - t], tau)

mu\_lz[i, s[i] - t] <- lz[i, s[i] - t + 1] - zr[i, s[i] - t + 1] - 1/(2 \* tau)

**Posterior Predictive Checks**

# Calculate squared residuals for model validation

y\_n\_sqs <- sum(y\_n\_sq[]) # Observed nest data

y\_n\_sqs\_new <- sum(y\_n\_sq\_new[]) # Simulated nest data

# Similar for initial and chick data...

**Derived Quantities**

# Annual growth multipliers

l\_a[i, t-1] <- exp(lz[i,t] - lz[i,t-1]) # Actual growth

l\_p[i, t-1] <- exp(zr[i,t]) # Process growth

# Geometric mean growth rates per site

gl\_a[i] <- pow(prod(lw\_a[i,]), (1/(sum(x[i,]) - 1)))

**3. Model Fitting & Parameter Extraction**

**MCMC Configuration**

**Parallel Chain Setup:**

r

n.chains <- 6 *# Number of parallel chains*

n.adapt <- 3000 *# Adaptation period*

n.update <- 300000 *# Burn-in iterations*

n.iter <- 200000 *# Sampling iterations*

thin <- 200 *# Thinning interval*

**Initialization Function:**

r

random\_inits <- function(model\_data) {

*# Generate starting values for all parameters*

*# - Random effects (eta, epsilon)*

*# - Abundance matrix (lz)*

*# - Breeding success (alpha)*

*# - Variance parameters (sigma\_site, sigma\_season, sigma)*

}

**Model Fitting Process**

**Parallel Execution:**

r

cl <- makeCluster(n.chains)

clusterExport(cl, c("model\_data", "random\_inits", ...))

out <- clusterEvalQ(cl, {

library(rjags)

inits <- random\_inits(model\_data)

*# Initialize model*

jm <- jags.model("jags\_model.jags",

data = model\_data,

inits = inits,

n.chains = 1,

n.adapt = n.adapt)

*# Burn-in*

update(jm, n.iter = n.update)

*# Sample posterior*

zm <- coda.samples(jm, variable.names = params,

n.iter = n.iter, thin = thin)

return(as.mcmc(zm))

})

model\_output <- mcmc.list(out)

**Parameter Extraction**

**Key Parameters Monitored:**

r

params <- c(

"beta", *# Growth rate intercept*

"sigma", *# Process variance*

"sigma\_site", *# Site-level variance*

"sigma\_season", *# Seasonal variance*

"alpha", *# Breeding success rates*

"epsilon", *# Seasonal effects*

"eta", *# Site effects*

"lz", *# Log abundances (primary output)*

"z\_c", *# Latent chick abundances*

"gl\_a", *# Geometric mean growth rates*

"l\_a", *# Annual growth multipliers*

*# Posterior predictive check variables*

"y\_i\_new", "y\_n\_new", "y\_c\_new",

"y\_n\_sqs", "y\_n\_sqs\_new", ...

)

**Abundance Extraction:**

r

*# Extract abundance estimates with uncertainty*

model\_samples <- as.matrix(model\_output)

lz\_columns <- grep("^lz\\[", colnames(model\_samples))

lz\_samples <- model\_samples[, lz\_columns]

*# Convert from log-scale to actual abundance*

abundance\_samples <- exp(lz\_samples)

*# Summarize with quantiles*

abundance\_summary <- apply(abundance\_samples, 2, function(x) {

c(mean = mean(x),

median = median(x),

lower\_95 = quantile(x, 0.025),

upper\_95 = quantile(x, 0.975))

})

**Extract Site and Season Indices:**

r

*# Parse parameter names to get site/season indices*

extract\_indices <- function(colname) {

indices <- gsub("[^0-9,]", "", colname)

as.integer(unlist(strsplit(indices, ",")))

}

indices <- lapply(colnames(abundance\_samples), extract\_indices)

sites <- sapply(indices, `[`, 1)

seasons <- sapply(indices, `[`, 2)

**Final Output Structure:**

r

final\_data <- data.frame(

site = sites,

season = seasons,

year = 1970 + seasons - 1,

mean\_abundance = abundance\_summary["mean", ],

median\_abundance = abundance\_summary["median", ],

lower\_95\_abundance = abundance\_summary["lower\_95.2.5%", ],

upper\_95\_abundance = abundance\_summary["upper\_95.97.5%", ]

) %>%

left\_join(SiteList, by = "site") %>% *# Add site metadata*

group\_by(site) %>%

mutate(growth\_rate = mean\_abundance / lag(mean\_abundance)) %>% *# Calculate growth*

ungroup()

**Model Outputs**

**Primary Results**

* **modeled\_gentoo\_parameters.csv**: Site-season abundance estimates with uncertainty
* **model\_data\_rinits\_output.rda**: Full MCMC posterior samples
* **random\_inits.rda**: Model initialization function

**Key Model Diagnostics**

* **Convergence**: Multiple chain diagnostics via coda package
* **Posterior Predictive Checks**: Observed vs. predicted data comparisons
* **Effective Sample Size**: Ensuring adequate posterior sampling

**Abundance Estimates Structure**

Each row represents a site-season combination with:

* Point estimates (mean, median)
* Uncertainty quantification (95% credible intervals)
* Annual growth rate calculations
* Site metadata (coordinates, names, etc.)

**Model Assumptions & Considerations**

**Critical Assumptions**

1. **Observation Error**: Log-normal with accuracy-dependent precision
2. **Process Model**: Random walk with drift on log-abundance scale
3. **Missing Data**: Presence/absence gap-filling rules
4. **Breeding Success**: Beta(a,b) distribution with mean=0.5, sd=0.25
5. **Random Effects**: Normal distributions for site and seasonal effects

**Model Limitations**

* **Detection**: Assumes perfect detection when present
* **Immigration/Emigration**: Not explicitly modeled
* **Environmental Covariates**: None in core abundance model
* **Temporal Autocorrelation**: Limited to AR(1) structure in random effects

**Validation Approaches**

* Posterior predictive checks compare observed vs. simulated data
* Cross-validation through temporal holdouts (not implemented)
* Sensitivity analysis of key assumptions (recommended addition)

**Computational Requirements**

**Minimum Specifications:**

* **RAM**: 16GB (32GB recommended for large datasets)
* **CPU**: 6+ cores for parallel MCMC
* **Runtime**: 4-8 hours depending on data size
* **Storage**: ~2GB for posterior samples

**Performance Tips:**

* Use parallel chains for faster convergence assessment
* Monitor convergence with gelman.diag() from coda