Urchin_Recruitment

Andrés Pinos-Sánchez

2025-05-15

##OBJECTIVE: #estimate the normalized year to year variance of incoming recruits

```
# Load libraries
library(nimble)
```

```
## Warning: package 'nimble' was built under R version 4.4.3
```

```
## nimble version 1.3.0 is loaded.
## For more information on NIMBLE and a User Manual,
## please visit https://R-nimble.org.
##
## Note for advanced users who have written their own MCMC samplers:
##   As of version 0.13.0, NIMBLE's protocol for handling posterior
##   predictive nodes has changed in a way that could affect user-defined
##   samplers in some situations. Please see Section 15.5.1 of the User Manual.
```

```
##
## Attaching package: 'nimble'
```

```
## The following object is masked from 'package:stats':
##
##     simulate
```

```
## The following object is masked from 'package:base':
##
##     declare
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.3
```

```
## Warning: package 'readr' was built under R version 4.4.3
```

```
## Warning: package 'forcats' was built under R version 4.4.3
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(mcmcplots)
```

```
## Warning: package 'mcmcplots' was built under R version 4.4.3
```

```
## Loading required package: coda
```

```
## Warning: package 'coda' was built under R version 4.4.3
```

```
library(MCMCvis)
```

```
## Warning: package 'MCMCvis' was built under R version 4.4.3
```

```
library(coda)

set.seed(123)

# Nimble
source("attach.nimble.R")

# Set working directory (Why not)
setwd("C:/Users/pinosa/OneDrive - Oregon State University/Ed_OSU_Thesis_GradSchool/OR_Trophic_Mo
del/Data/PISCO_Recruitment_Data")
```

##CLEAN DATA

```
# PISCO data
pisco <- read_csv("PISCO_all_years.csv") # source("PISCO_data_prep.R")
```

```
## Rows: 77630 Columns: 15
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr  (7): sample_month, site_code, exposure, zone, collector_type, sampler, ...
## dbl  (6): year, replicate, proportion_sampled, count_classcode, count, metho...
## date (2): deploy_date, collect_date
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# 1. Filter for taxa 66,67,70 and drop unwanted sites
urchins_raw <- pisco %>%
  filter(count_classcode %in% c("66","67","70"), # data 2001 - 2011
         !site_code %in% c("CMEN00","CMES00","IFBRXX","KHLX00",
                           "PSGX00","SHCX00","TRHX00","IPTAXX")) # 14 sites
```

##NON BAYESIAN METHOD (CHANGED CODE FROM JESS R CODE)

Results: Normalized variance across years: 0.0706 Normalized SD across years: 0.2657

```
# 2. SUM WITHIN “TRANSECTS” ----
# here we treat each (year, site, zone, replicate) as one transect
transect_totals <- urchins_raw %>%
  group_by(year, site_code, zone, replicate) %>%
  summarise(count = sum(count), .groups="drop")

# 3. ANNUAL MEAN + SD ----
annual_stats <- transect_totals %>%
  group_by(year) %>%
  summarise(
    mean_count = mean(count),
    sd_count   = sd(count),
    .groups = "drop"
  )

# 4. LOG-TRANSFORM (for log-normal modeling) ----
# add +1 pseudocount to handle zeros safely
annual_stats <- annual_stats %>%
  mutate(log_mean = log(mean_count + 1))

# 5. DETREND ON THE LOG SCALE ----
trend_mod <- lm(log_mean ~ year, data = annual_stats)

annual_stats <- annual_stats %>%
  mutate(
    # add back the overall mean of log_mean to the residuals
    detrended_log = mean(log_mean) + resid(trend_mod)
  )

# 6. BACK–TRANSFORM & NORMALIZE ----
annual_stats <- annual_stats %>%
  mutate(
    detrended = exp(detrended_log),            # back to count scale
    normalized = detrended / mean(detrended)   # unit mean
  )

# 7. EXTRACT NORMALIZED VARIANCE & SD ----
norm_var <- var(annual_stats$normalized)
norm_sd  <- sd(annual_stats$normalized)

# print results:
cat("Normalized variance across years:", round(norm_var, 4), "\n")
```

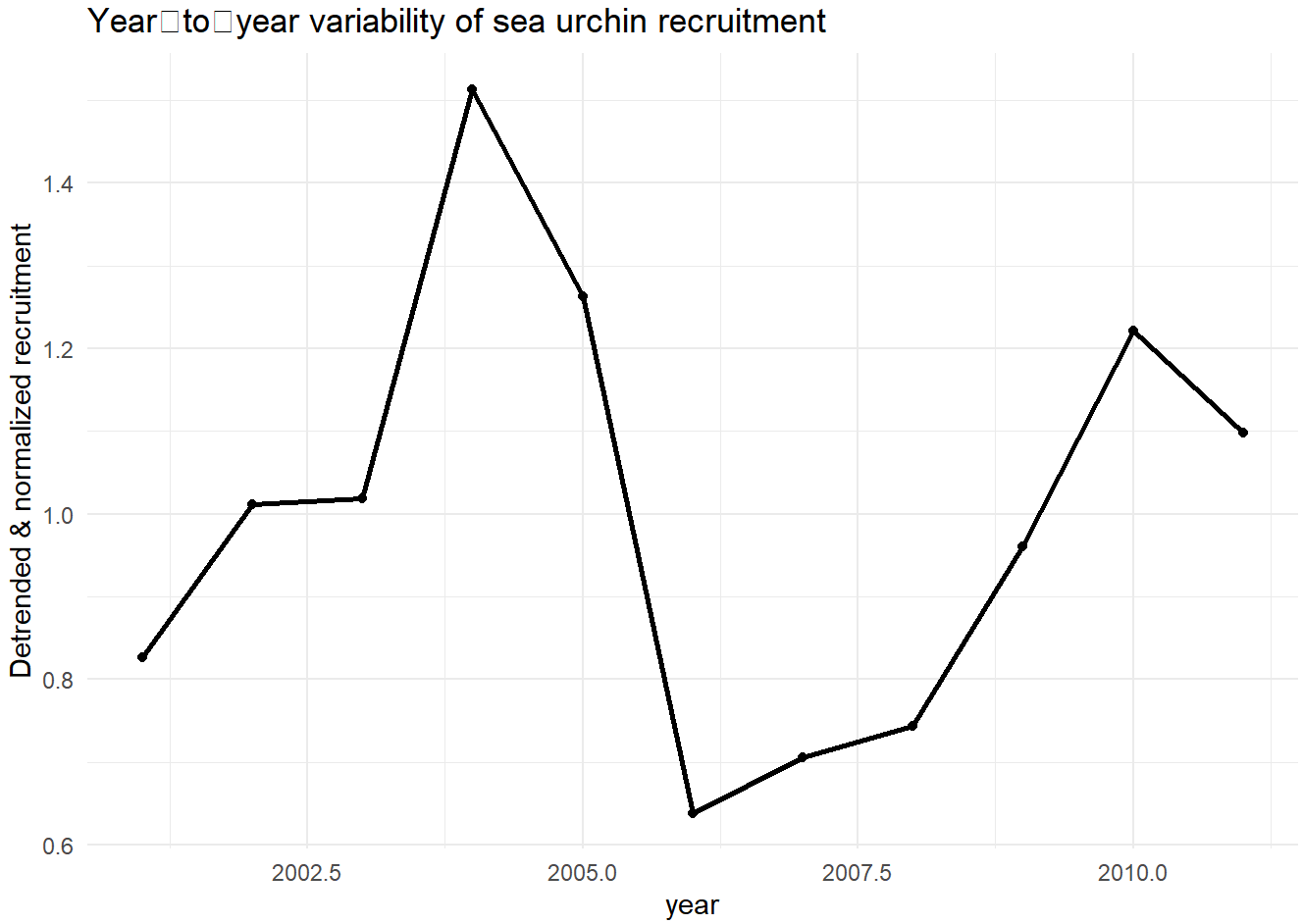
```
## Normalized variance across years: 0.0706
```

```
cat("Normalized SD across years:",       round(norm_sd, 4), "\n")
```

```
## Normalized SD across years: 0.2657
```

```
# 8. PLOT: detrended and normalized time-series ----
library(ggplot2)
ggplot(annual_stats, aes(x = year, y = normalized)) +
  geom_line(size = 1) +
  geom_point() +
  labs(
    y = "Detrended & normalized recruitment",
    title = "Year-to-year variability of sea urchin recruitment"
  ) +
  theme_minimal()
```

```
## 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.
```



##BAYESIAN METHOD Normalized variance across years: 0.08110644 Normalized SD across years: 0.11129504

```
# 2. group by year & sum within transects
# here we treat each (year, site, zone, replicate) as one transect
transect_totals <- urchins_raw %>%
  group_by(year, site_code, zone, replicate) %>% #zone, replicate???
  summarise(count = sum(count), .groups="drop")

# 3. Compute annual mean recruitment
annual_stats <- transect_totals %>%
  group_by(year) %>%
  summarise(mean_count = mean(count),
            sd_count   = sd(count),
            .groups    = "drop") %>%
  # log-transform with +1 pseudocount
  mutate(log_y  = log(mean_count + 1), # center year for numerical stability
         year_c = year - mean(year))

# 4. Nimble code (Let's make Josh proud)
nim_code <- nimbleCode({
  # priors for trend intercept and slope
  alpha ~ dnorm(0, sd = 10)
  beta  ~ dnorm(0, sd =  1)

  # prior on residual precision -> convert to SD
  tau   ~ dgamma(0.001, 0.001)
  sigma <- 1 / sqrt(tau)

  # likelihood: log-mean around trend
  for(i in 1:N.year) {

    # process model
    mu[i]    <- alpha + beta * year_c[i]

    # observation model
    log_y[i] ~ dnorm(mu[i], tau = tau)

  }#i

  # derived: normalized variance on original scale
  var_norm <- exp(sigma * sigma) - 1 #Var(X)/E(X)^2 = exp(σ^2) - 1 for a lognormal

})#nim_code

# 5. Data prep + nimble constants
parameters <- c("alpha", "beta", "sigma", "var_norm")

nimble.data <- list(log_y  = annual_stats$log_y)

nimble.constants <- list(N.year = nrow(annual_stats),
                         year_c = annual_stats$year_c)

n_iter    <- 20000
n_burnin  <- 2000
```

```
n_chains  <- 3
n_thin    <- 10

# 6. Run MCMC
mcmc.output <- nimbleMCMC(code       = nim_code,
                          data       = nimble.data,
                          constants  = nimble.constants,
                          monitors   = parameters,
                          niter      = n_iter,
                          nburnin    = n_burnin,
                          thin       = n_thin,
                          nchains    = n_chains,
                          summary    = TRUE,
                          samplesAsCodaMCMC = TRUE)
```

```
## Defining model
```

```
## Building model
```

```
## Setting data and initial values
```

```
## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model variables.
```

```
## Checking model sizes and dimensions
```

```
##   [Note] This model is not fully initialized. This is not an error.
##          To see which variables are not initialized, use model$initializeInfo().
##          For more information on model initialization, see help(modelInitialization).
```

```
## Checking model calculations
```

```
## [Note] NAs were detected in model variables: alpha, logProb_alpha, beta, logProb_beta, tau, l
ogProb_tau, mu, sigma, lifted_d1_over_sqrt_oPtau_cP, var_norm, logProb_log_y.
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
## running chain 2...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
## running chain 3...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
# attach sample objects (alpha, beta, sigma, var_norm, etc.)
attach.nimble(mcmc.output$samples)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
## The following object is masked from 'package:nimble':
##
##     cube
```

```
# 7. Summarize & Plot
summary(mcmc.output)
```

```
##         Length Class     Mode
## samples 3      mcmc.list list
## summary 4      -none-    list
```

```
mcmcplot(mcmc.output$samples)
```

```
##                                                                          Preparing plo
ts for alpha.  25% complete.
```

```
##                                                                          Preparing plo
ts for beta.  50% complete.
```

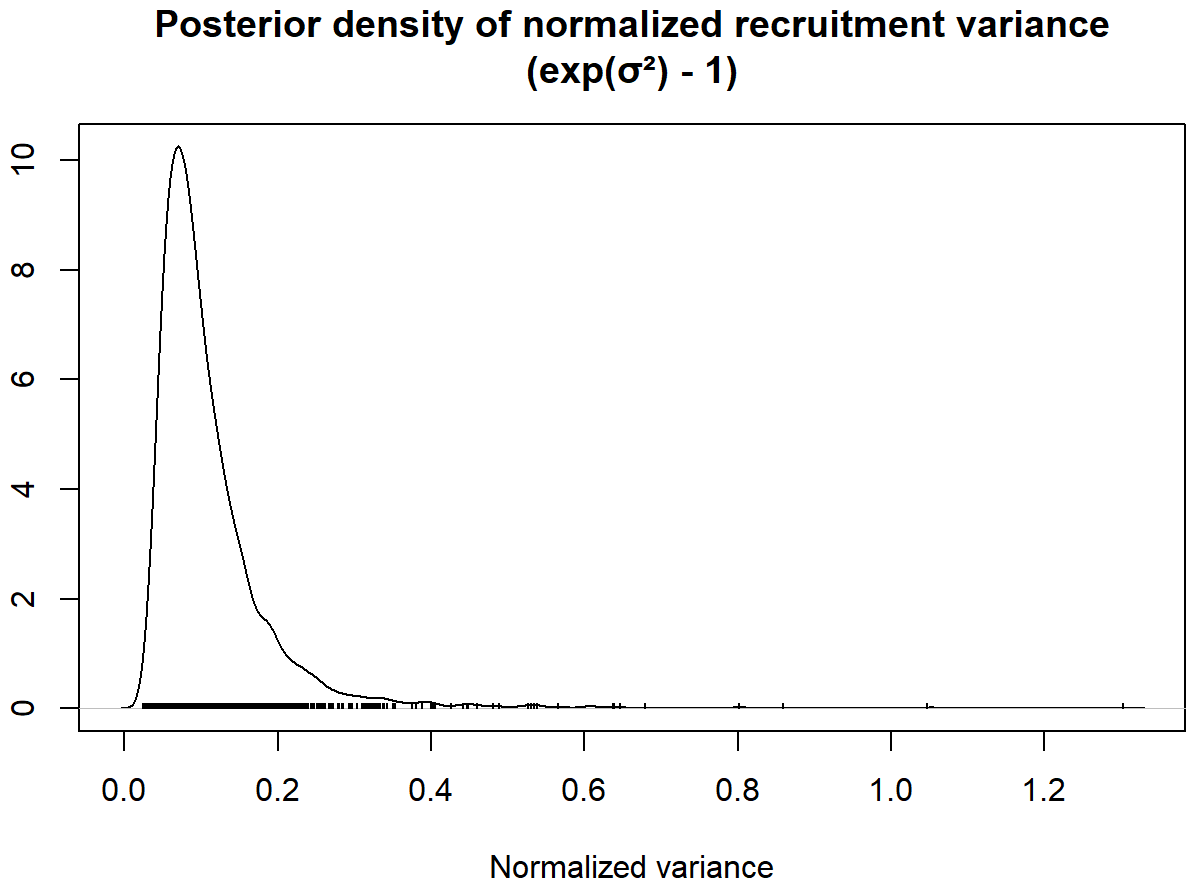
```
##                                                                          Preparing plo
ts for sigma.  75% complete.
```

```
##                                                                          Preparing plo
ts for var.  100% complete.
```

```
print(mcmc.output$summary)
```

```
## $chain1
##                Mean      Median    St.Dev.   95%CI_low   95%CI_upp
## alpha     0.4329758  0.43117588 0.09419134  0.24749996  0.61500600
## beta     -0.0841521 -0.08379090 0.03120273 -0.14718015 -0.02207027
## sigma     0.3080968  0.29168190 0.08469743  0.19292022  0.52275965
## var_norm  0.1100801  0.08880235 0.08315858  0.03791948  0.31426512
##
## $chain2
##                 Mean      Median    St.Dev.  95%CI_low   95%CI_upp
## alpha     0.43044038  0.42790518 0.09853833  0.2273451  0.63336779
## beta     -0.08280644 -0.08348451 0.03026155 -0.1424243 -0.02308421
## sigma     0.30855208  0.29119625 0.08617047  0.1913651  0.51167314
## var_norm  0.11065887  0.08849418 0.08190874  0.0372994  0.29927890
##
## $chain3
##                 Mean      Median    St.Dev.   95%CI_low   95%CI_upp
## alpha     0.42955212  0.42902130 0.09914383  0.23314786  0.62826501
## beta     -0.08494921 -0.08426259 0.03223784 -0.15268460 -0.02060907
## sigma     0.31309746  0.29750175 0.08368918  0.19817309  0.52436453
## var_norm  0.11314616  0.09254222 0.07818168  0.04005401  0.31647567
##
## $all.chains
##                 Mean      Median    St.Dev.   95%CI_low   95%CI_upp
## alpha     0.43098944  0.42936577 0.09730895  0.23449660  0.62434022
## beta     -0.08396925 -0.08391539 0.03125119 -0.14707965 -0.02230696
## sigma     0.30991545  0.29340086 0.08487280  0.19362677  0.52145153
## var_norm  0.11129504  0.08989795 0.08110644  0.03820299  0.31247111
```

```
# Plot posterior density of var_norm
densplot(mcmc.output$samples[, "var_norm"],
         main = "Posterior density of normalized recruitment variance\n(exp(σ²) - 1)",
         xlab = "Normalized variance")
```




```
# RESULT: normalize variance = 0.111295
```