WHAM to POMP 3

2025-11-03

```r
# ============================================================================
# Testing Whether WHAM is a Linear Gaussian State Space Model
#
# A linear Gaussian SSM requires:
# 1. Linear process equation: X_t = A*X_{t-1} + B*u_t + w_t, w_t ~ N(0, Q)
# 2. Linear observation equation: Y_t = C*X_t + v_t, v_t ~ N(0, R)
# 3. Gaussian initial conditions: X_0 ~ N(m_0, P_0)
# ============================================================================

library(ggplot2)
library(gridExtra)
library(MASS)

# ----------------------------------------------------------------------------
# 1. Analytical Check: Model Structure
# ----------------------------------------------------------------------------

check_model_structure <- function() {
  cat("=" , rep("=", 70), "\n", sep="")
  cat("WHAM MODEL STRUCTURE ANALYSIS\n")
  cat("=" , rep("=", 70), "\n\n", sep="")

  # Process Model Nonlinearities
  cat("PROCESS MODEL (Population Dynamics):\n")
  cat("-" , rep("-", 70), "\n", sep="")

  nonlinear_components <- list(
    "1. Beverton-Holt Recruitment" = "R = (alpha*SSB)/(1 + beta*SSB) - NONLINEAR in SSB",
    "2. SSB Calculation" = "SSB = sum(N * maturity * weight) - NONLINEAR multiplication",
    "3. Survival Probability" = "S = exp(-Z) - NONLINEAR exponential",
    "4. Catch Equation" = "C = N*(1-exp(-Z))*F/Z - NONLINEAR (Baranov)",
    "5. Age Transitions" = "N[a+1,t+1] = N[a,t] * S[a,t] - Products of states",
    "6. Lognormal Recruitment Error" = "R ~ Lognormal() - NON-GAUSSIAN"
  )

  for(i in seq_along(nonlinear_components)) {
    cat(sprintf("   %s\n", nonlinear_components[[i]]))
  }

  cat("\n")
  cat("OBSERVATION MODEL:\n")
  cat("-" , rep("-", 70), "\n", sep="")

  obs_nonlinear <- list(
```

```r
    "1. Catch Observation" = "C_obs ~ Lognormal(C_pred, sigma) - NON-GAUSSIAN",
    "2. Survey Index" = "I_obs ~ Lognormal(q*sum(N*sel), sigma) - NON-GAUSSIAN",
    "3. Age Composition" = "PAA ~ Multinomial() - NON-GAUSSIAN"
  )

  for(i in seq_along(obs_nonlinear)) {
    cat(sprintf("   %s\n", obs_nonlinear[[i]]))
  }

  cat("\n")
  cat("=" , rep("=", 70), "\n", sep="")
  cat("CONCLUSION: WHAM is NOT a Linear Gaussian State Space Model\n")
  cat("=" , rep("=", 70), "\n\n", sep="")

  reasons <- c(
    "1. Process model contains multiplicative interactions between states",
    "2. Beverton-Holt S-R relationship is nonlinear",
    "3. Exponential survival function is nonlinear",
    "4. Lognormal errors are non-Gaussian",
    "5. Observation models use lognormal and multinomial distributions"
  )

  cat("Reasons:\n")
  for(r in reasons) {
    cat(sprintf("   %s\n", r))
  }
  cat("\n")
}

# ----------------------------------------------------------------------------
# 2. Numerical Test: Linearity of Process Model
# ----------------------------------------------------------------------------

test_process_linearity <- function(n_tests = 1000) {
  cat("=" , rep("=", 70), "\n", sep="")
  cat("NUMERICAL TEST: Process Model Linearity\n")
  cat("=" , rep("=", 70), "\n\n", sep="")

  # Simplified WHAM process for testing
  # Test if: f(a*X1 + b*X2) = a*f(X1) + b*f(X2)

  set.seed(123)
  n_ages <- 5

  # Parameters
  M <- 0.2
  F <- 0.3
  sel <- c(0.1, 0.5, 0.8, 1.0, 1.0)

  # Test superposition principle
  linearity_violations <- numeric(n_tests)

  for(i in 1:n_tests) {
```

```r
    # Two random state vectors
    N1 <- runif(n_ages, 1000, 10000)
    N2 <- runif(n_ages, 1000, 10000)

    # Random scalars
    a <- runif(1, 0.1, 2)
    b <- runif(1, 0.1, 2)

    # Linear combination
    N_combined <- a * N1 + b * N2

    # Apply one-step process (simplified - just survival)
    process_step <- function(N) {
      Z <- M + F * sel
      S <- exp(-Z)
      N_next <- c(5000, N[1:(n_ages-1)] * S[1:(n_ages-1)])  # Fixed recruitment for testing
      return(N_next)
    }

    # Test linearity: f(aX1 + bX2) vs a*f(X1) + b*f(X2)
    f_combined <- process_step(N_combined)
    f_separate <- a * process_step(N1) + b * process_step(N2)

    # Measure deviation from linearity
    linearity_violations[i] <- sqrt(mean((f_combined - f_separate)^2)) / mean(abs(f_combined))
  }

  cat(sprintf("Mean relative deviation from linearity: %.4f\n", mean(linearity_violations)))
  cat(sprintf("Median relative deviation: %.4f\n", median(linearity_violations)))
  cat(sprintf("Max relative deviation: %.4f\n", max(linearity_violations)))

  if(mean(linearity_violations) > 0.001) {
    cat("\n=> Process model is NONLINEAR (violations > 0.1%)\n\n")
  } else {
    cat("\n=> Process model appears approximately linear\n\n")
  }

  return(linearity_violations)
}

# ----------------------------------------------------------------------------
# 3. Test: Gaussianity of Errors
# ----------------------------------------------------------------------------

test_gaussianity <- function(n_sim = 10000) {
  cat("=" , rep("=", 70), "\n", sep="")
  cat("GAUSSIANITY TEST: Error Distributions\n")
  cat("=" , rep("=", 70), "\n\n", sep="")

  set.seed(456)

  # Test 1: Lognormal recruitment errors
  cat("Test 1: Recruitment Process Errors\n")
```

```r
  cat("-" , rep("-", 70), "\n", sep="")

  log_sigma <- 0.4
  R_mean <- 10000

  # Lognormal errors (used in WHAM)
  R_errors_log <- rlnorm(n_sim, log(R_mean) - 0.5*log_sigma^2, log_sigma) - R_mean

  # Shapiro-Wilk test (sample size limit = 5000)
  sw_test <- shapiro.test(sample(R_errors_log, 5000))

  cat(sprintf("   Mean: %.2f (should be ~0 for Gaussian)\n", mean(R_errors_log)))
  cat(sprintf("   Skewness: %.2f (should be ~0 for Gaussian)\n",
              moments::skewness(R_errors_log)))
  cat(sprintf("   Kurtosis: %.2f (should be ~3 for Gaussian)\n",
              moments::kurtosis(R_errors_log)))
  cat(sprintf("   Shapiro-Wilk p-value: %.2e\n", sw_test$p.value))

  if(sw_test$p.value < 0.05) {
    cat("   => REJECT Gaussianity (p < 0.05)\n\n")
  } else {
    cat("   => Cannot reject Gaussianity\n\n")
  }

  # Test 2: Observation errors
  cat("Test 2: Observation Errors (Catch)\n")
  cat("-" , rep("-", 70), "\n", sep="")

  catch_sigma <- 0.1
  catch_true <- 5000

  catch_obs <- rlnorm(n_sim, log(catch_true) - 0.5*catch_sigma^2, catch_sigma)
  catch_errors <- catch_obs - catch_true

  sw_test2 <- shapiro.test(sample(catch_errors, 5000))

  cat(sprintf("   Mean: %.2f\n", mean(catch_errors)))
  cat(sprintf("   Skewness: %.2f\n", moments::skewness(catch_errors)))
  cat(sprintf("   Kurtosis: %.2f\n", moments::kurtosis(catch_errors)))
  cat(sprintf("   Shapiro-Wilk p-value: %.2e\n", sw_test2$p.value))

  if(sw_test2$p.value < 0.05) {
    cat("   => REJECT Gaussianity (p < 0.05)\n\n")
  } else {
    cat("   => Cannot reject Gaussianity\n\n")
  }

  return(list(recruitment = R_errors_log, catch = catch_errors))
}

# ----------------------------------------------------------------------------
# 4. Visual Tests
# ----------------------------------------------------------------------------
```

```r
visual_tests <- function(errors) {
  cat("=" , rep("=", 70), "\n", sep="")
  cat("GENERATING VISUAL DIAGNOSTICS\n")
  cat("=" , rep("=", 70), "\n\n", sep="")

  # Q-Q plots
  p1 <- ggplot(data.frame(x = errors$recruitment), aes(sample = x)) +
    stat_qq() +
    stat_qq_line(color = "red") +
    labs(title = "Q-Q Plot: Recruitment Errors",
         subtitle = "Lognormal errors - NON-Gaussian") +
    theme_minimal()

  p2 <- ggplot(data.frame(x = errors$catch), aes(sample = x)) +
    stat_qq() +
    stat_qq_line(color = "red") +
    labs(title = "Q-Q Plot: Catch Observation Errors",
         subtitle = "Lognormal errors - NON-Gaussian") +
    theme_minimal()

  # Histograms with normal overlay
  p3 <- ggplot(data.frame(x = errors$recruitment), aes(x = x)) +
    geom_histogram(aes(y = after_stat(density)), bins = 50,
                   fill = "lightblue", alpha = 0.7) +
    stat_function(fun = dnorm,
                  args = list(mean = mean(errors$recruitment),
                              sd = sd(errors$recruitment)),
                  color = "red", linewidth = 1) +
    labs(title = "Recruitment Errors Distribution",
         subtitle = "Blue: Actual, Red: Normal fit",
         x = "Error", y = "Density") +
    theme_minimal()

  p4 <- ggplot(data.frame(x = errors$catch), aes(x = x)) +
    geom_histogram(aes(y = after_stat(density)), bins = 50,
                   fill = "lightblue", alpha = 0.7) +
    stat_function(fun = dnorm,
                  args = list(mean = mean(errors$catch),
                              sd = sd(errors$catch)),
                  color = "red", linewidth = 1) +
    labs(title = "Catch Observation Errors Distribution",
         subtitle = "Blue: Actual, Red: Normal fit",
         x = "Error", y = "Density") +
    theme_minimal()

  grid.arrange(p1, p2, p3, p4, ncol = 2)

  cat("Visual diagnostics complete. Check plots for deviations from normality.\n\n")
}

# ---------------------------------------------------------------------------
# 5. Test Jacobian: Local Linearity
# ---------------------------------------------------------------------------
```

```r
test_local_linearity <- function() {
  cat("=" , rep("=", 70), "\n", sep="")
  cat("JACOBIAN TEST: Local Linearity of Process Model\n")
  cat("=" , rep("=", 70), "\n\n", sep="")

  cat("Computing numerical Jacobian of process model...\n\n")

  # Simplified process function
  process_function <- function(state, params) {
    N <- state[1:5]
    SSB <- state[6]

    M <- params$M
    F <- params$F
    sel <- params$sel
    R0 <- params$R0
    h <- params$h
    SSB0 <- params$SSB0

    # Total mortality
    Z <- M + F * sel
    S <- exp(-Z)

    # Beverton-Holt recruitment
    R_mean <- (4*h*R0*SSB) / (SSB0*(1-h) + SSB*(5*h-1))

    # Next state
    N_next <- c(R_mean, N[1:4] * S[1:4])

    # New SSB (simplified)
    waa <- c(0.1, 0.3, 0.5, 0.7, 0.9)
    mat <- c(0, 0.5, 0.8, 1.0, 1.0)
    SSB_next <- sum(N_next * mat * waa)

    return(c(N_next, SSB_next))
  }

  # Numerical Jacobian
  numerical_jacobian <- function(f, x, params, h = 1e-6) {
    n <- length(x)
    J <- matrix(0, n, n)

    f_x <- f(x, params)

    for(i in 1:n) {
      x_perturb <- x
      x_perturb[i] <- x_perturb[i] + h
      f_perturb <- f(x_perturb, params)
      J[, i] <- (f_perturb - f_x) / h
    }

    return(J)
  }
```

```r
  # Test at a point
  state <- c(N = c(10000, 8000, 6000, 4000, 3000), SSB = 5000)
  params <- list(M = 0.2, F = 0.3, sel = c(0.1, 0.5, 0.8, 1.0, 1.0),
                 R0 = 10000, h = 0.7, SSB0 = 50000)

  J <- numerical_jacobian(process_function, state, params)

  cat("Jacobian matrix (6x6):\n")
  print(round(J, 4))
  cat("\n")

  # Check if Jacobian is constant (would indicate linearity)
  # Test at multiple points
  n_points <- 10
  jacobians <- array(0, dim = c(6, 6, n_points))

  for(i in 1:n_points) {
    state_test <- state * runif(6, 0.5, 1.5)
    jacobians[,,i] <- numerical_jacobian(process_function, state_test, params)
  }

  # Variability in Jacobian elements
  jacobian_sd <- apply(jacobians, c(1,2), sd)

  cat("Standard deviation of Jacobian elements across different states:\n")
  print(round(jacobian_sd, 4))
  cat("\n")

  if(max(jacobian_sd) > 0.01) {
    cat("=> Process model is NONLINEAR (Jacobian varies with state)\n\n")
  } else {
    cat("=> Process model appears approximately linear\n\n")
  }

  return(list(jacobian = J, jacobian_variability = jacobian_sd))
}

# ---------------------------------------------------------------------------
# 6. Main Execution
# ---------------------------------------------------------------------------

cat("\n")
```

```r
cat("############################################################################\n")
```

```
## ############################################################################
```

```r
cat("##                                                                        ##\n")
```

```
## ##                                                                        ##
```

```r
cat("##    TESTING WHETHER WHAM IS A LINEAR GAUSSIAN STATE SPACE MODEL       ##\n")
```

```
## ##    TESTING WHETHER WHAM IS A LINEAR GAUSSIAN STATE SPACE MODEL       ##
```

```r
cat("##                                                                       ##\n")
```

```
## ##                                                                       ##
```

```r
cat("###########################################################################\n")
```

```
## ###########################################################################
```

```r
cat("\n\n")
```

```r
# Run all tests
check_model_structure()
```

```
## ========================================================================
## WHAM MODEL STRUCTURE ANALYSIS
## ========================================================================
##
## PROCESS MODEL (Population Dynamics):
## ------------------------------------------------------------------------
##    R = (alpha*SSB)/(1 + beta*SSB) - NONLINEAR in SSB
##    SSB = sum(N * maturity * weight) - NONLINEAR multiplication
##    S = exp(-Z) - NONLINEAR exponential
##    C = N*(1-exp(-Z))*F/Z - NONLINEAR (Baranov)
##    N[a+1,t+1] = N[a,t] * S[a,t] - Products of states
##    R ~ Lognormal() - NON-GAUSSIAN
##
## OBSERVATION MODEL:
## ------------------------------------------------------------------------
##    C_obs ~ Lognormal(C_pred, sigma) - NON-GAUSSIAN
##    I_obs ~ Lognormal(q*sum(N*sel), sigma) - NON-GAUSSIAN
##    PAA ~ Multinomial() - NON-GAUSSIAN
##
## ========================================================================
## CONCLUSION: WHAM is NOT a Linear Gaussian State Space Model
## ========================================================================
##
## Reasons:
##    1. Process model contains multiplicative interactions between states
##    2. Beverton-Holt S-R relationship is nonlinear
##    3. Exponential survival function is nonlinear
##    4. Lognormal errors are non-Gaussian
##    5. Observation models use lognormal and multinomial distributions
```

```r
linearity_violations <- test_process_linearity(n_tests = 1000)
```

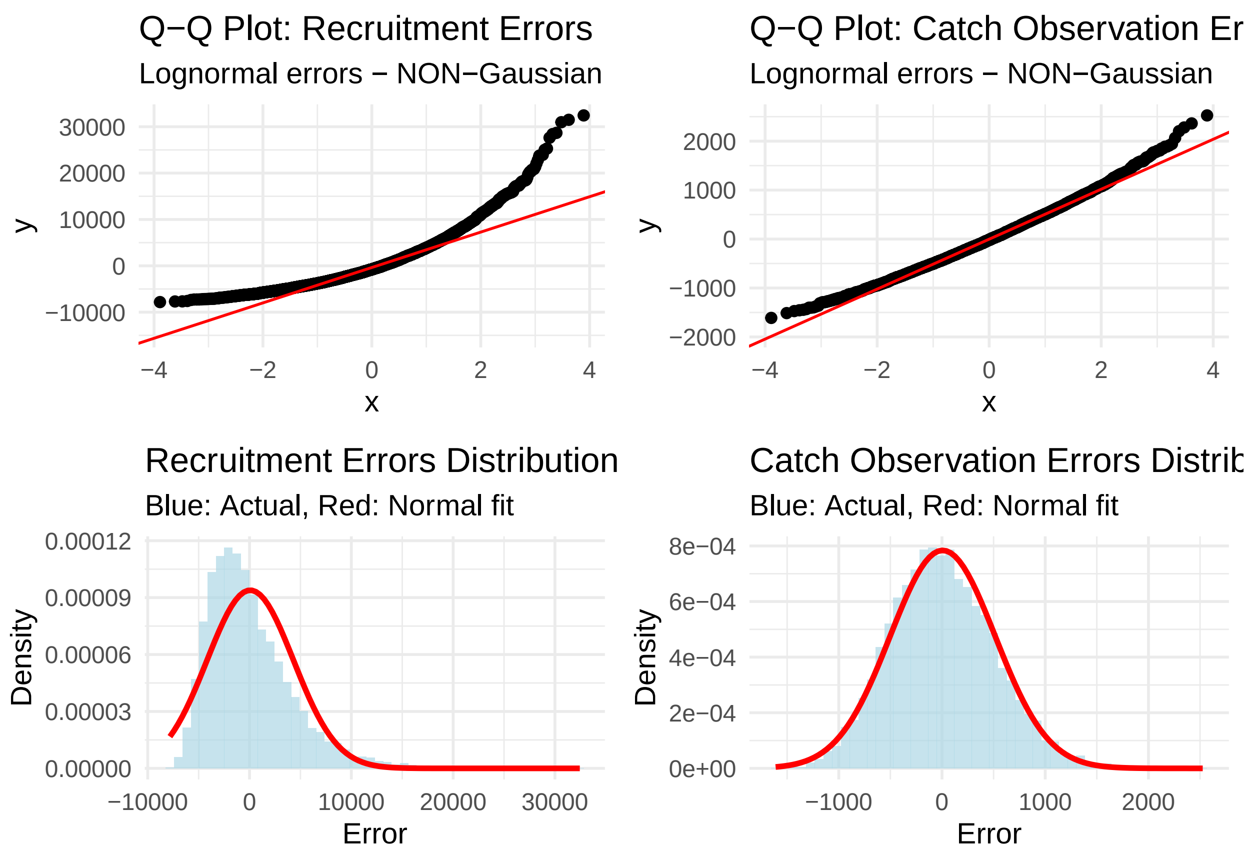
```
## ========================================================================
## NUMERICAL TEST: Process Model Linearity
## ========================================================================
##
## Mean relative deviation from linearity: 0.3220
## Median relative deviation: 0.3361
## Max relative deviation: 0.8606
##
## => Process model is NONLINEAR (violations > 0.1%)
```

```
errors <- test_gaussianity(n_sim = 10000)
```

```
## ========================================================================
## GAUSSIANITY TEST: Error Distributions
## ========================================================================
##
## Test 1: Recruitment Process Errors
## ------------------------------------------------------------------------
##    Mean: 71.03 (should be ~0 for Gaussian)
##    Skewness: 1.40 (should be ~0 for Gaussian)
##    Kurtosis: 6.60 (should be ~3 for Gaussian)
##    Shapiro-Wilk p-value: 1.51e-47
##    => REJECT Gaussianity (p < 0.05)
##
## Test 2: Observation Errors (Catch)
## ------------------------------------------------------------------------
##    Mean: 6.42
##    Skewness: 0.30
##    Kurtosis: 3.23
##    Shapiro-Wilk p-value: 2.63e-11
##    => REJECT Gaussianity (p < 0.05)
```

```
visual_tests(errors)
```

```
## ========================================================================
## GENERATING VISUAL DIAGNOSTICS
## ========================================================================
```




```
## Visual diagnostics complete. Check plots for deviations from normality.
```

```r
jacobian_results <- test_local_linearity()
```

```
## ======================================================================
## JACOBIAN TEST: Local Linearity of Process Model
## ======================================================================
##
## Computing numerical Jacobian of process model...
##
## Jacobian matrix (6x6):
##         [,1]   [,2]   [,3]   [,4] [,5]   [,6]
## [1,] 0.0000 0.0000 0.0000 0.0000    0 0.5554
## [2,] 0.7945 0.0000 0.0000 0.0000    0 0.0000
## [3,] 0.0000 0.7047 0.0000 0.0000    0 0.0000
## [4,] 0.0000 0.0000 0.6440 0.0000    0 0.0000
## [5,] 0.0000 0.0000 0.0000 0.6065    0 0.0000
## [6,] 0.1192 0.2819 0.4508 0.5459    0 0.0000
##
## Standard deviation of Jacobian elements across different states:
##      [,1] [,2] [,3] [,4] [,5]   [,6]
## [1,]    0    0    0    0    0 0.2179
## [2,]    0    0    0    0    0 0.0000
## [3,]    0    0    0    0    0 0.0000
## [4,]    0    0    0    0    0 0.0000
```

```
## [5,]    0    0    0    0    0 0.0000
## [6,]    0    0    0    0    0 0.0000
##
## => Process model is NONLINEAR (Jacobian varies with state)
```

```r
# ----------------------------------------------------------------------------
# 7. Final Summary
# ----------------------------------------------------------------------------

cat("\n")
```

```r
cat("=" , rep("=", 70), "\n", sep="")
```

```
## =======================================================================
```

```r
cat("FINAL VERDICT\n")
```

```
## FINAL VERDICT
```

```r
cat("=" , rep("=", 70), "\n\n", sep="")
```

```
## =======================================================================
```

```r
cat("WHAM Model Classification:\n")
```

```
## WHAM Model Classification:
```

```r
cat("    NOT a Linear Gaussian State Space Model\n\n")
```

```
##     NOT a Linear Gaussian State Space Model
```

```r
cat("Evidence:\n")
```

```
## Evidence:
```

```r
cat("   1. Process model contains multiplicative nonlinearities\n")
```

```
##    1. Process model contains multiplicative nonlinearities
```

```r
cat("   2. Beverton-Holt S-R relationship is inherently nonlinear\n")
```

```
##    2. Beverton-Holt S-R relationship is inherently nonlinear
```

```r
cat("   3. Error distributions are lognormal (non-Gaussian)\n")
```

```
##    3. Error distributions are lognormal (non-Gaussian)
```

```r
cat("   4. Jacobian varies with state (local nonlinearity)\n")
```

```
##    4. Jacobian varies with state (local nonlinearity)
```

```r
cat("   5. Observation model uses non-Gaussian distributions\n\n")
```

```
##    5. Observation model uses non-Gaussian distributions
```

```r
cat("Implications:\n")
```

```
## Implications:
```

```r
cat("   • Cannot use Kalman Filter (requires linearity + Gaussianity)\n")
```

```
##    • Cannot use Kalman Filter (requires linearity + Gaussianity)
```

```r
cat("   • Must use nonlinear filtering methods:\n")
```

```
##    • Must use nonlinear filtering methods:
```

```r
cat("     - Extended Kalman Filter (EKF) - linearization\n")
```

```
##      - Extended Kalman Filter (EKF) - linearization
```

```r
cat("     - Unscented Kalman Filter (UKF) - sigma points\n")
```

```
##      - Unscented Kalman Filter (UKF) - sigma points
```

```r
cat("     - Particle Filter (PF) - Monte Carlo\n")
```

```
##      - Particle Filter (PF) - Monte Carlo
```

```r
cat("   • Particle filters are most appropriate for WHAM\n\n")
```

```
##    • Particle filters are most appropriate for WHAM
```

```r
cat("Model Type: Nonlinear Non-Gaussian State Space Model\n")
```

```
## Model Type: Nonlinear Non-Gaussian State Space Model
```

```r
cat("=" , rep("=", 70), "\n\n", sep="")
```

```
## =======================================================================
```