Part II: Basic penultimate Weibull–XIMIS implementation

Alison Peard

2025-10-21

The following functions are adapted from the SI of Cook (2023).

Define XIMIS reduced variate and quantile estimators. The `yximis` plotting position estimator was derived in Harris (2009). There is one set of plotting positions for every dataset of size M exceedances over a record length of R (e.g., R years).

```
# plotting positions
yximis <- function(M = 100, R = 1) {
  y <- rep(0, M)
  var <- y
  y[1] <- -digamma(1) + log(R)  # Euler's constant = -digamma(1) = 0.5772...
  var[1] <- pi^2 / 6

  if (M > 1) {
    for (i in 2:M) {
      y[i] <- y[i-1] - 1/(i-1)
      var[i] <- var[i-1] - 1/(i-1)^2
    }
  }

  data.frame(mean = y, var = var)
}
```

The XIMIS quantile function is defined as,

$$\hat{V} = (U^w + \hat{y} D^w)^{\frac{1}{w}}.$$

So for a given mean recurrence interval (MRI),

$$\hat{y}_{\mathrm{MRI}} = -\log\left(-\log\left(1 - \frac{1}{\mathrm{MRI}}\right)\right).$$

```
qximis <- function(y, U, D, w) {
  (U^w + y * D^w)^(1/w)
}
```

XIMIS fitting functions (weighted least mean squares) with and without fitting the tail index too.

```
# XIMIS fit with known tail index
pot.XMS <- function(V, R = length(V), w = 1) {
  x <- sort(V^w, decreasing = TRUE)
  y <- yximis(length(x), R) # get reduced variate
  my <- y$mean
  wt <- 1 / y$var
```

```
  lm(x ~ my, weights = wt) # fit for U and D
}

# XIMIS by weighted MLS with free fit of Weibull index w
pot.XMW <- function(V, R = length(V)) {
  M <- length(V)
  y <- yximis(M, R)$mean
  var <- yximis(M, R)$var

  fit <- pot.XMS(V, R, 1)
  par <- c(coef(fit)[1], coef(fit)[2], 1)

  errfn <- function(x) {
    sum((V - qximis(y, par[1], par[2], par[3]))^2/var)
  }
  optim(par, errfn)
}
```

Now define simulation parameters.

```
# Weibull parameters
w_true <- 2          # Shape parameter (typical for synoptic winds)
C <- 10              # Scale parameter
r <- 22             # Rate of independent peaks per epoch (year)
R <- 16              # Number of epochs (years of data)
M <- 30             # Number of POT values to use

# Return level to predict
MRI <- 50            # 50-year return level

# Number of bootstrap trials
n_trials <- 3 # 1000
```

Generate POT data from a Weibull parent.

```
set.seed(42)  # For reproducibility

# Generate parent Weibull data
N <- r * R  # Total population
parent_data <- C * (-log(runif(N)))^(1/w_true) # equiv to rweibull(N, shape = w_true, scale = C)

# Select top M values (POT approach)
V <- sort(parent_data, decreasing = TRUE)[1:M]

cat("Generated POT data:\n")
```

```
## Generated POT data:
```

```
cat(sprintf("  Number of POT values (M): %d\n", M))
```

```
##   Number of POT values (M): 30
```

```
cat(sprintf("  Record length (R): %d epochs\n", R))
```

```
##   Record length (R): 16 epochs
```

```r
cat(sprintf("  Total samples (N): %d\n", N))
```

```
##   Total samples (N): 352
```

```r
cat(sprintf("  Rate per epoch (r): %d\n", r))
```

```
##   Rate per epoch (r): 22
```

```r
cat(sprintf("  Top value: %.2f\n", max(V)))
```

```
##   Top value: 28.88
```

```r
cat(sprintf("  Threshold (M-th value): %.2f\n\n", min(V)))
```

```
##   Threshold (M-th value): 15.68
```

Fit the XIMIS model.

```r
# Fit XIMIS with known w
fit <- pot.XMS(V, R, w = w_true)

# Extract parameters (note: these are in transformed space V^w)
U_w <- coef(fit)[1]
D_w <- coef(fit)[2]

# Transform back to original space
U <- U_w^(1/w_true)
D <- D_w^(1/w_true)

cat("XIMIS fitted parameters:\n")
```

```
## XIMIS fitted parameters:
```

```r
cat(sprintf("  Mode (U): %.2f\n", U))
```

```
##   Mode (U): 18.29
```

```r
cat(sprintf("  Dispersion (D): %.2f\n", D))
```

```
##   Dispersion (D): 12.50
```

```r
cat(sprintf("  Shape (w): %.2f (fixed)\n\n", w_true))
```

```
##   Shape (w): 2.00 (fixed)
```

Predict the 50-year return level.

```r
# Calculate reduced variate for MRI
y_50 <- -log(-log(1 - 1/MRI))

# Predict 50-year return level
V_50_pred <- qximis(y_50, U, D, w_true)

# True 50-year value from source Weibull
# For Weibull with rate r, the mode U = C^w * ln(r)
U_true <- C^w_true * log(r)
D_true <- C^w_true
V_50_true <- (U_true + y_50 * D_true)^(1/w_true)

cat("50-year return level predictions:\n")
```

```
## 50-year return level predictions:
```

```r
cat(sprintf("  True value: %.2f\n", V_50_true))
```

```
##   True value: 26.44
```

```r
cat(sprintf("  XIMIS prediction: %.2f\n", V_50_pred))
```

```
##   XIMIS prediction: 30.73
```

```r
cat(sprintf("  Error: %.2f%%\n\n", 100 * (V_50_pred - V_50_true) / V_50_true))
```

```
##   Error: 16.21%
```

Bootstrap analysis (multiple trials).

```r
cat(sprintf("Running %d bootstrap trials...\n", n_trials))
```

```
## Running 3 bootstrap trials...
```

```r
predictions <- numeric(n_trials)

for (i in 1:n_trials) {
  # Generate new sample
  parent <- C * (-log(runif(N)))^(1/w_true)
  V_trial <- sort(parent, decreasing = TRUE)[1:M]

  # Fit XIMIS
  fit_trial <- pot.XMS(V_trial, R, w = w_true)
  U_trial <- coef(fit_trial)[1]^(1/w_true)
  D_trial <- coef(fit_trial)[2]^(1/w_true)

  # Predict
  predictions[i] <- qximis(y_50, U_trial, D_trial, w_true)
}
```

Results summary.

```r
cat("\nBootstrap results (", n_trials, "trials):\n", sep = "")
```

```
##
## Bootstrap results (3trials):
```

```r
cat(sprintf("  True 50-year value: %.2f\n", V_50_true))
```

```
##   True 50-year value: 26.44
```

```r
cat(sprintf("  Mean prediction: %.2f\n", mean(predictions)))
```

```
##   Mean prediction: 26.38
```

```r
cat(sprintf("  Std. dev: %.2f\n", sd(predictions)))
```

```
##   Std. dev: 0.71
```

```r
cat(sprintf("  Standard error: %.2f%%\n", 100 * sd(predictions) / V_50_true))
```

```
##   Standard error: 2.68%
```

```r
cat(sprintf("  Bias: %.2f%%\n", 100 * (mean(predictions) - V_50_true) / V_50_true))
```

```
##   Bias: -0.24%
```

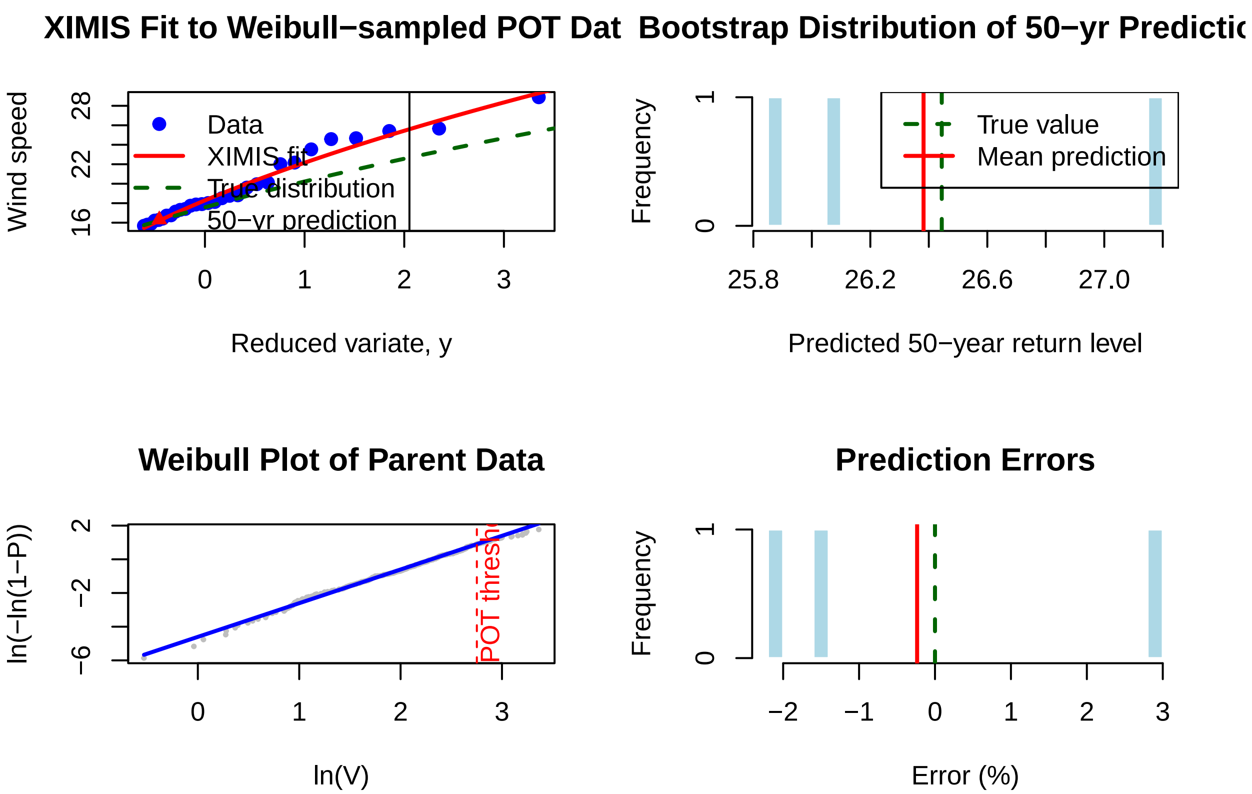
```r
cat(sprintf("  95%% CI: [%.2f, %.2f]\n",
    quantile(predictions, 0.025),
    quantile(predictions, 0.975)))
```

```
##   95% CI: [25.90, 27.14]
```

Figures

```r
par(mfrow = c(2, 2))

# 1. Gumbel plot of data with XIMIS fit
y_vals <- yximis(M, R)$mean
plot(y_vals, sort(V, decreasing = TRUE),
     xlab = "Reduced variate, y",
     ylab = "Wind speed",
     main = "XIMIS Fit to Weibull-sampled POT Data",
     pch = 19, col = "blue")

# Add fitted line
y_seq <- seq(min(y_vals), max(y_vals) + 2, length.out = 100)
V_fit <- qximis(y_seq, U, D, w_true)
lines(y_seq, V_fit, col = "red", lwd = 2)

# Add 50-year prediction
points(y_50, V_50_pred, pch = 17, col = "red", cex = 1.5)
text(y_50, V_50_pred, labels = "50-yr", pos = 4, col = "red")

# Add true distribution
V_true_line <- (U_true + y_seq * D_true)^(1/w_true)
lines(y_seq, V_true_line, col = "darkgreen", lwd = 2, lty = 2)

legend("topleft",
       legend = c("Data", "XIMIS fit", "True distribution", "50-yr prediction"),
       col = c("blue", "red", "darkgreen", "red"),
       pch = c(19, NA, NA, 17),
       lty = c(NA, 1, 2, NA),
       lwd = c(NA, 2, 2, NA))

# 2. Bootstrap distribution
hist(predictions, breaks = 30,
     main = "Bootstrap Distribution of 50-yr Predictions",
     xlab = "Predicted 50-year return level",
     col = "lightblue", border = "white")
abline(v = V_50_true, col = "darkgreen", lwd = 2, lty = 2)
abline(v = mean(predictions), col = "red", lwd = 2)
legend("topright",
       legend = c("True value", "Mean prediction"),
       col = c("darkgreen", "red"),
       lwd = 2, lty = c(2, 1))

# 3. Weibull plot of parent data
P_weibull <- (1:N) / (N + 1)
plot(log(parent_data[order(parent_data)]), log(-log(1 - P_weibull)),
     xlab = "ln(V)", ylab = "ln(-ln(1-P))",
     main = "Weibull Plot of Parent Data",
```

```r
      pch = 19, cex = 0.3, col = "gray")

# Add theoretical line
V_seq <- seq(min(parent_data), max(parent_data), length.out = 100)
P_theory <- 1 - exp(-(V_seq/C)^w_true)
lines(log(V_seq), log(-log(1 - P_theory)), col = "blue", lwd = 2)

# Highlight POT threshold
abline(v = log(min(V)), col = "red", lty = 2)
text(log(min(V)), par("usr")[3], labels = "POT threshold",
     pos = 4, col = "red", srt = 90)

# 4. Prediction errors
errors <- (predictions - V_50_true) / V_50_true * 100
hist(errors, breaks = 30,
     main = "Prediction Errors",
     xlab = "Error (%)",
     col = "lightblue", border = "white")
abline(v = 0, col = "darkgreen", lwd = 2, lty = 2)
abline(v = mean(errors), col = "red", lwd = 2)
```




```r
par(mfrow = c(1, 1))
```

ADDITIONAL: Compare with different MRIs

```r
MRIs <- c(10, 20, 50, 100, 500, 1000, 10000)
y_MRIs <- -log(-log(1 - 1/MRIs))

predictions_matrix <- matrix(0, nrow = n_trials, ncol = length(MRIs))

for (i in 1:n_trials) {
  parent <- C * (-log(runif(N)))^(1/w_true)
  V_trial <- sort(parent, decreasing = TRUE)[1:M]
  fit_trial <- pot.XMS(V_trial, R, w = w_true)
  U_trial <- coef(fit_trial)[1]^(1/w_true)
  D_trial <- coef(fit_trial)[2]^(1/w_true)

  for (j in 1:length(MRIs)) {
    predictions_matrix[i, j] <- qximis(y_MRIs[j], U_trial, D_trial, w_true)
  }
}

# Calculate standard errors
std_errors <- apply(predictions_matrix, 2, sd)
true_values <- (U_true + y_MRIs * D_true)^(1/w_true)
relative_errors <- 100 * std_errors / true_values

cat("\nStandard errors by return period:\n")
```

```
##
## Standard errors by return period:
```

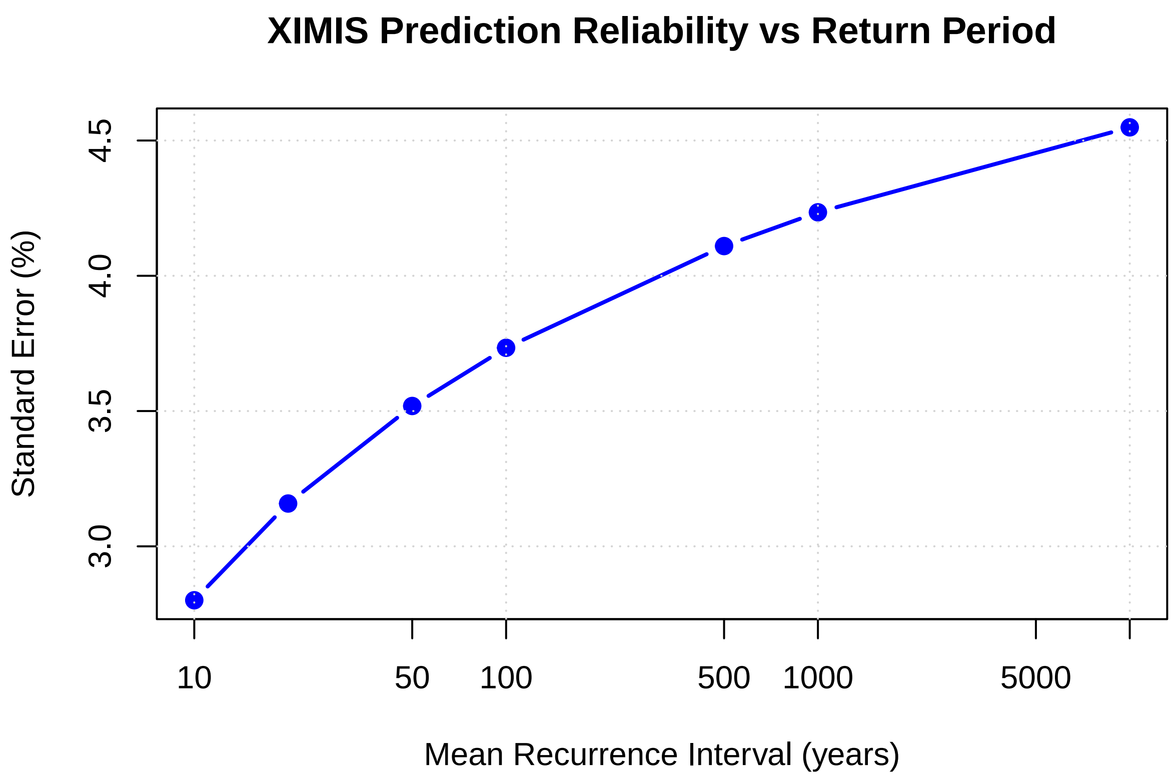
```r
cat(sprintf("%8s %12s %12s\n", "MRI", "Std Error", "Rel Error (%)"))
```

```
##      MRI    Std Error Rel Error (%)
```

```r
for (i in 1:length(MRIs)) {
  cat(sprintf("%8d %12.2f %12.2f\n", MRIs[i], std_errors[i], relative_errors[i]))
}
```

```
##       10         0.65         2.80
##       20         0.78         3.16
##       50         0.93         3.52
##      100         1.04         3.73
##      500         1.25         4.11
##     1000         1.34         4.23
##    10000         1.60         4.55
```

```r
plot(MRIs, relative_errors, type = "b", log = "x",
     xlab = "Mean Recurrence Interval (years)",
     ylab = "Standard Error (%)",
     main = "XIMIS Prediction Reliability vs Return Period",
     pch = 19, col = "blue", lwd = 2)
grid()
```


XIMIS Prediction Reliability vs Return Period
Standard Error (%)
3.0
3.5
4.0
4.5
10
50
100
500
1000
5000
Mean Recurrence Interval (years)