KRONX: Clock of Regimes in R

Oscar Linares

November 5, 2026

Abstract

This vignette introduces the `KRONX` package for regime-switching fragility analysis. The package extends a standard Hidden Markov Model workflow by constructing an open operator chain

$$A \to Q \to K \to N,$$

where Q is a hazard-adjusted transition operator, $K = Q - I$ is a sub-generator, and $N = -K^{-1}$ is the fundamental matrix of cumulative expected residence times. The package provides a compact workflow for fitting Gaussian and Student-t hidden Markov models, constructing the KRONX operators, and computing a finite-horizon ruin bound.

Contents

1 Introduction **2**

2 Package overview **2**
2.1 Dependencies 2
2.2 Core operator chain 3

3 Quick start **3**

4 Main functions **3**
4.1 `run_kronx()` 3
4.2 `fit_hmm_em()` 4
4.3 `viterbi_decode()` 5
4.4 `read_close_series()` and `compute_log_returns()` 5

5 Worked example **5**
5.1 Constructing the operator chain 5
5.2 Risk-bound components 6
5.3 Optional file-based workflow 6

6 Testing **6**

7 Conclusion **6**

1 Introduction

The `KRONX` package implements the Clock of Regimes framework as an open-system extension of a Hidden Markov Model. In a standard HMM, persistence is encoded through one-step transition probabilities a_{ii}. In KRONX, the estimated transition structure is converted into a survival-style operator system:

$$A \to Q \to K \to N.$$

Starting from a fitted transition matrix A, the package assigns state-specific hazards ϵ_i, constructs

$$Q = \mathrm{diag}(1 - \epsilon_i)A,$$

then defines

$$K = Q - I, \qquad N = -K^{-1}.$$

The matrix N summarizes cumulative expected sojourn times before absorption and provides the residence-time geometry underlying the KRONX interpretation.

2 Package overview

The package supports the following workflow:

1. read a price series or accept a numeric return vector,
2. fit Gaussian and Student-t HMMs,
3. reorder states by scale,
4. decode the latent state path,
5. construct Q, K, and N,
6. compute quasi-stationary and residence weights,
7. compute a finite-horizon ruin bound.

2.1 Dependencies

- Requires: R $\geq$ 4.0.0
- Imports: `stats`, `utils`
- Suggests: `testthat` $\geq$ 3.0.0
- No compiled code

2.2 Core operator chain

Symbol	Definition	Interpretation
Q	$\mathrm{diag}(1-\varepsilon)A$	Leaky transition operator
K	$Q - I$	Sub-generator
N	$-K^{-1}$	Fundamental matrix of cumulative sojourn times

Table 1: KRONX operator chain.

3 Quick start

For package documentation, the safest approach is a small self-contained example.

```
> set.seed(123)
> x <- rnorm(120, mean = 0, sd = 0.01)
> res <- KRONX::run_kronx(
+   x = x,
+   K = 2L,
+   n_starts = 1L,
+   max_iter = 10L,
+   tol = 1e-4,
+   tail_alpha = 0.05,
+   ruin_horizon = 25L,
+   verbose = FALSE,
+   output_dir = NULL
+ )
> res$ruin_bound

[1] 0.01926856

> dim(res$A_t)

[1] 2 2
```

The object returned by `run_kronx()` contains the fitted HMMs, the KRONX operator chain, and summary risk quantities.

4 Main functions

The package exports:

```
> getNamespaceExports("KRONX")
```

4.1 `run_kronx()`

`run_kronx()` is the main entry point. It accepts either a file path or a numeric return vector.

```
> KRONX::run_kronx(
+   file = NULL,
+   x = NULL,
+   close_col = "close",
+   K = 3L,
+   n_starts = 5L,
+   max_iter = 200L,
+   tol = 1e-6,
+   seed = 123L,
+   epsilon_min = 0.01,
+   c_hazard = 0.05,
+   tail_alpha = 0.01,
+   ruin_horizon = 250L,
+   nu_grid = c(3:30, 40, 60, 100),
+   verbose = TRUE,
+   output_dir = "kronx_output"
+ )
```

Its workflow is:

1. ingest data or accept a supplied return vector,
2. fit Gaussian and Student-t HMMs,
3. decode the Student-t state path,
4. build ϵ, Q, K, and N,
5. compute quasi-stationary weights, residence weights, and the ruin bound,
6. optionally write output files.

4.2 `fit_hmm_em()`

`fit_hmm_em()` fits either a Gaussian or Student-t HMM by EM.

```
> set.seed(123)
> x <- rnorm(100, 0, 0.01)
> fit <- KRONX::fit_hmm_em(
+   x = x,
+   K = 2L,
+   model = "gaussian",
+   n_starts = 1L,
+   max_iter = 8L,
+   tol = 1e-4,
+   verbose = FALSE
+ )
> fit$logLik

[1] 328.2374
```

4.3 `viterbi_decode()`

```
> states <- KRONX::viterbi_decode(x, fit)
> table(states)

states
  2
100
```

4.4 `read_close_series()` and `compute_log_returns()`

These functions support data ingestion and return construction.

```
> px <- KRONX::read_close_series("my_prices.csv", close_col = "close")
> ret <- KRONX::compute_log_returns(px)
> summary(ret)
```

5 Worked example

5.1 Constructing the operator chain

```
> set.seed(123)
> x <- rnorm(120, 0, 0.01)
> t_fit <- KRONX::fit_hmm_em(
+   x = x,
+   K = 2L,
+   model = "student",
+   n_starts = 1L,
+   max_iter = 8L,
+   tol = 1e-4,
+   nu_grid = c(3, 5, 10),
+   verbose = FALSE
+ )
> epsilon <- KRONX:::hazard_from_nu(
+   nu = t_fit$params$nu,
+   epsilon_min = 0.01,
+   c_hazard = 0.05
+ )
> Q <- KRONX:::build_Q(t_fit$A, epsilon)
> K_mat <- KRONX:::build_K(Q)
> N_mat <- KRONX:::fundamental_matrix(K_mat)
> epsilon

[1] 0.015 0.015

> Q

          [,1]      [,2]
[1,] 0.8532308 0.1317692
[2,] 0.0711881 0.9138119
```

5.2 Risk-bound components

```
> loss_threshold <- stats::quantile(x, probs = 0.05)
> q_state <- KRONX:::left_tail_state_probs(t_fit, loss_threshold)
> pi_qs <- KRONX:::quasi_stationary_distribution(Q)
> pi_res <- KRONX:::residence_weight_vector(N_mat, init = pi_qs)
> bar_q <- sum(pi_res * q_state)
> bar_lambda <- sum(pi_res * epsilon)
> ruin_bound <- 1 - exp(-bar_lambda * bar_q * 25)
> c(bar_q = bar_q, bar_lambda = bar_lambda, ruin_bound = ruin_bound)

     bar_q bar_lambda ruin_bound
0.07274146 0.01500000 0.02690936
```

5.3 Optional file-based workflow

For larger real-data workflows, use a CSV file with a close-price column.

```
> res_file <- KRONX::run_kronx(
+   file = "IBKR_ES_2Y1h.csv",
+   close_col = "close",
+   verbose = FALSE,
+   output_dir = NULL
+ )
> res_file$ruin_bound
```

6 Testing

The package includes `testthat` tests covering:

- end-to-end workflow on toy data,
- file-based ingestion checks,
- input validation,
- HMM fitting interface validation,
- operator-chain structural properties.

Tests can be run with:

```
> devtools::test()
> testthat::test_package("KRONX")
```

7 Conclusion

The `KRONX` package provides a complete regime-switching fragility workflow in pure R. It combines HMM estimation with an open operator framework based on Q, K, and N. For package purposes, this vignette focuses on the computational workflow and a small reproducible example. Longer theoretical and journal-style exposition is better maintained as a separate paper or preprint.