

Package: coxstream (via r-universe)

June 20, 2026

Title Memory-Efficient Cox Proportional Hazards via Streaming
Newton-Raphson

Version 0.1.0

Description Fits the Cox proportional hazards model using a single descending-order pass per Newton-Raphson iteration. Peak RAM is $O(p^2)$ regardless of the number of rows, making it suitable for datasets that do not fit in memory. Produces identical coefficients to `survival::coxph()` with Efron tie correction.

URL <https://github.com/tommycarstensen/coxstream-r>

BugReports <https://github.com/tommycarstensen/coxstream-r/issues>

License MIT + file LICENSE

Encoding UTF-8

Imports Rcpp, survival

LinkingTo Rcpp

Suggests arrow, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Tommy Carstensen [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-3672-9931>>), Apache Software
Foundation [cph, ctb] (vendored Arrow C Data/Stream interface
header (src/arrow_c_abi.h), Apache-2.0)

Maintainer Tommy Carstensen <cran@tommycarstensen.com>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-06-20 13:40:11 UTC

RemoteUrl <https://github.com/cran/coxstream>

RemoteRef HEAD

RemoteSha 7b774d05726a7fec35232b001456a1896155c011

Contents

coxstream	2
coxstream_arrow	3

Index	5
--------------	----------

coxstream	<i>Fit a Cox proportional hazards model via streaming Newton-Raphson</i>
-----------	--

Description

Fits the Cox PH model using a single descending-time-order pass per Newton-Raphson iteration. Peak RAM is $O(p^2)$ regardless of n , making it suitable for large datasets. Produces identical coefficients to `survival::coxph()` with Efron tie correction.

Usage

```
coxstream(
  formula,
  data,
  init = NULL,
  max_iter = 25L,
  tol = 1e-09,
  verbose = FALSE
)
```

Arguments

formula	A formula with a <code>survival::Surv()</code> response, e.g. <code>Surv(time, event) ~ x1 + x2</code> .
data	A data frame containing the variables in formula.
init	Optional numeric vector of starting values for beta (length p). Defaults to zero.
max_iter	Maximum Newton-Raphson iterations. Default 25.
tol	Convergence tolerance on the max absolute score element. Default 1e-9.
verbose	Currently unused; reserved for future per-iteration output. Default FALSE.

Value

An object of class "coxstream" with components:

coefficients	Named numeric vector of fitted coefficients.
var	Variance-covariance matrix (inverse of observed information).
loglik	Log-likelihood at convergence.
n_iter	Number of NR iterations taken.
n	Number of rows.
formula	The formula used.
call	The matched call.

Examples

```
library(survival)
fit <- coxstream(Surv(time, status) ~ age + sex, data = lung)
coef(fit)
```

coxstream_arrow	<i>Fit a Cox PH model by streaming a DESC-sorted parquet file</i>
-----------------	---

Description

Like `coxstream()` but reads data row-group by row-group from parquet. Peak RAM is $O(\text{batch_size} * p)$ for the active chunk plus $O(p^2)$ for the carry state, independent of total n . Uses exact Efron tie correction: tie groups that span row-group boundaries are handled via local carry state, giving bit-identical coefficients to `coxstream()` on any data.

Usage

```
coxstream_arrow(
  parquet_path,
  x_cols,
  time_col = "duration",
  event_col = "event",
  init = NULL,
  max_iter = 25L,
  tol = 1e-08,
  batch_size = 250000L,
  verbose = TRUE
)
```

Arguments

<code>parquet_path</code>	Path to a parquet file sorted by time DESCENDING.
<code>x_cols</code>	Character vector of covariate column names.
<code>time_col</code>	Column name for event/censoring time. Default "duration".
<code>event_col</code>	Column name for event indicator (1 = event). Default "event".
<code>init</code>	Optional starting values for beta (length p). Default zero.
<code>max_iter</code>	Maximum NR iterations. Default 25.
<code>tol</code>	Convergence tolerance on $\ NR\text{ step}\ $ (L2 norm of beta update). Default 1e-8. Same criterion as the Python <code>coxstream</code> implementations.
<code>batch_size</code>	Target rows per read call. Consecutive row groups are merged until the total reaches this size, then freed (with a <code>gc()</code>) before the next is read, so peak RAM is $O(\text{batch_size} * p)$, flat in n . The default 250 000 keeps RAM genuinely flat; larger chunks are slightly faster but let the allocator's high-water ratchet up, so RAM regains a mild upward drift.
<code>verbose</code>	Print per-iteration progress. Default TRUE.

Details

Each NR iteration reads one row-group chunk at a time with `mmap = FALSE` (pread into heap buffers freed after each chunk – a memory-mapped reader would instead leave every touched file page resident for the mapping's lifetime, making RSS grow $O(n)$). Each chunk is exported to a C ArrowArrayStream and consumed zero-copy in C++ by `efron_stream_chunk_inplace()`, with the Efron tie-state carried across chunks in R – no R-level column materialisation (`as.vector / cbind / concat_tables`), which is what previously left a $\sim 1.5x$ gap behind the Python streaming path.

Value

A "coxstream" object (same class as `coxstream()`).

Index

coxstream, [2](#)

coxstream_arrow, [3](#)