Archimedean Copula Inference via Taylor-Mode AD

Cambridge Yang
Independent
research@camyang.com

Dongdong Li
Harvard Medical School
dongdongli@hsph.harvard.edu

Abstract

No existing nested Archimedean copula tool handles all three of (a) arbitrary per-variable (right-)censoring in survival analysis, (b) arbitrary nesting trees, and (c) exact parameter gradients. Existing implementations handle only bivariate problems, low dimensional (i.e., $d \leq 10$) cases, two layers of nesting, or only hand-derived copula nestings. We present ACOPULA, a JAX-native framework that, given any Archimedean generator—classical or neural—evaluates exact nested-copula likelihoods and parameter gradients under arbitrary censoring masks in polynomial time. The mechanism is polynomial powering of Taylor-mode automatic differentiation output, which replaces per-family hand-derived partial Bell polynomial tables with a single differentiable computation that any user-defined generator can drive. We conduct extensive simulations to verify the correctness of ACOPULA. We then demonstrate (a) per-variable censoring on $85{,}229$ MIMIC-IV ICU admissions in high dimensions with d=53, fit by both classical Archimedean families and nested neural Archimedean copulas; (b) an 11-sector hierarchical model on S&P 500 daily returns at d=98; (c) family-agnostic censored MLE across ten copula families, five of them with no prior implementation, on a retinopathy study; and (d) a $\sim$650$\times$ per-density speedup over R's `nacLL` at d=35, scaling quadratically to d=8,000.

1 Introduction

Hierarchical multivariate data with partial observation are common in modern survival, financial, and clinical risk settings. Patients in a survival study with paired organs may never experience the event during follow-up [Huster et al., 1989]. Firms in a credit portfolio default in industry-sector clusters [Hofert, 2010], and a firm that survives the observation window has a right-censored default time. Intensive care unit (ICU) lab measurements group by clinical panel, and the time until each lab value first becomes abnormal is right-censored at ICU discharge [Johnson et al., 2023]. Each of these settings requires a model that captures the hierarchy with interpretable parameters (e.g., tail dependence and concordance) and allows each observation to have a different subset of censored variables.

Archimedean copulas. A *copula* $C : [0,1]^d \to [0,1]$ is a multivariate distribution on the unit cube with d uniform marginals; by Sklar's theorem [Sklar, 1959], every joint distribution on $\mathbf{u} = (u_1, \ldots, u_d) \in [0,1]^d$ factors uniquely into its one-dimensional marginals and a copula that captures their dependence. This decomposition lets the analyst pair domain-specific marginal models—e.g., Weibull [Weibull, 1951] for survival times, GARCH [Bollerslev, 1986] for returns—with an interpretable dependence structure whose parameters directly quantify tail behaviour and concordance. *Nested Archimedean copulas* [McNeil, 2008, Hofert and Pham, 2013] extend copulas to hierarchical dependence by composing generator functions in a tree: variables under the same internal node share stronger dependence than variables across branches.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

The inference barrier. Fitting a nested copula requires its density—a mixed partial derivative of the copula cumulative distribution function (CDF), one differentiation per observed variable—through a tree of composed generators [Hofert and Pham, 2013]; existing tools face three challenges.

- *Arbitrary nesting and any generator.* R's `copula` package [Hofert et al., 2012] hard-codes per-family Stirling-number tables for five generators and supports two nesting layers; adding a sixth family—or any neural-based generator [Ling et al., 2020, Liu et al., 2025b]—requires fresh hand derivation. The recent Julia package Copulas.jl [Laverny and Jimenez, 2024] evaluates single-layer Archimedean copula densities by Taylor-mode automatic differentiation (AD) over the generator, and therefore admits arbitrary generators, but did not support nesting until we ported our algorithm into it (Appendix K).
- *Per-variable censoring.* In the survival setting we consider, each variable of an observation can be right-censored separately, and each set of censored variables changes which mixed partial derivative of the copula CDF defines the likelihood, so for a d=53 ICU dataset with thousands of distinct censoring patterns the framework must handle every censoring mask $\delta = (\delta_1, \ldots, \delta_d) \in \{0, 1\}^d$, where $\delta_j = 1$ if variable $j \in \{1, \ldots, d\}$ is observed and $\delta_j = 0$ if it is censored. Prior tools cover bivariate problems [Sun and Ding, 2020, Emura and Chen, 2018] or shared censoring [Liu et al., 2025b], tested up to d=6 (Table 1).
- *Exact automatic parameter gradients.* Without automatic gradients, fitting a high-dimensional model (e.g., d=53 in Section 5.3) relies on derivative-free or finite-difference methods that scale poorly, or on Monte Carlo approximations [Hofert et al., 2013, Ng et al., 2021] whose accuracy depends on the number of Monte Carlo samples (Appendix R).

Our approach. We present ACOPULA, a JAX-native framework that resolves the three challenges. The user specifies at minimum a copula generator and ACOPULA computes the nested copula likelihood for any tree shape and any censoring mask, with exact parameter gradients automatically. The mechanism is polynomial powering of Taylor coefficients: Taylor-mode AD [Griewank and Walther, 2008, Radul et al., 2023] returns the higher-order derivatives of each generator composition in one forward pass, and raising those coefficient vectors to successive powers via convolution recovers the partial Bell polynomials that the density formula needs without per-family hand derivation, for any generator satisfying the standard d_c-monotonicity nesting condition [McNeil, 2008].

Contributions. The contributions of this paper are fivefold:

1. A *family-agnostic, gradient-based likelihood inference algorithm* for nested Archimedean copulas: given only the generator (e.g., a classical family or a *Deep Archimedean Copula* [Ling et al., 2020, Liu et al., 2025b]), the framework derives densities and gradients automatically.
2. *Per-variable censoring*: the likelihood algorithm handles any subset of censored variables in each observation within the same recursion, whereas prior censoring-aware tools are restricted to bivariate data or to one censoring time shared by all variables.
3. *Complexity results*: the algorithm costs $O(d^3)$ in the worst case for trees of fixed depth and $O(d^2)$ for $\sqrt{d}$*-decomposed* trees of fixed depth, in which every subtree below the root contains at most $O(\sqrt{d})$ uncensored variables. We validate these results empirically up to d=8,000.
4. An *open-source JAX implementation*, ACOPULA, with a simple API for nested-copula density and gradients, plus a per-edge d_c-monotonicity diagnostic for the nesting-validity condition.
5. As a by-product, conditional-distribution sampling [Hofert, 2008] of these copulas, which needs the same high-order derivatives, becomes practical. We defer it to Appendix F due to space.

Related work. We position ACOPULA against four related lines of work; Table 1 compares their capabilities.

- *Classical ecosystem.* The R `copula` [Hofert et al., 2012, 2013] and HAC [Okhrin and Ristig, 2014] packages compute generator derivatives via hand-coded closed forms for five families. The simulated maximum likelihood estimator relies on Laplace-transform frailty sampling, with Monte Carlo variance that grows with dimension; our exact-likelihood approach attains $\sim$100$\times$ speedup at comparable accuracy (Appendix R), and our deterministic-density comparison against R's `nacLL` reaches $\sim$650$\times$ at d=35 (Section 5.2).
- *Censoring-aware copula models.* CopulaCenR [Sun and Ding, 2020] fits censored bivariate and trivariate copulas, including interval-censoring via 2^d-corner inclusion–exclusion of the survival copula; Li et al. [2020, 2023] handle right-censoring using brute-force symbolic differentiation.

Table 1: Capability comparison across tools. Copulas.jl entries reflect commit `aeb7611c`, before our nesting port (Appendix K); N.I.: framework admits feature but currently not implemented.

	R `copula`	HAC	Copulas.jl	HACSurv†	DCSurvival	CopulaCenR	Vine	**Ours**
User-defined classical generator	×	×	✓	N/A	×	×	N/A	✓
Deep Archimedean generators	×	×	N.I.	✓	✓	×	N/A	✓
Sampling	✓	✓	✓	N/A	✓	✓	✓	✓
Nesting depth (exact density)	2	any	1	2	N/A	N/A	N/A	any
Automatic density gradients	×	×	✓	✓	✓	×	×	✓
Family-agnostic density	×	×	✓	×	✓	N/A	×	✓
Per-variable censoring	none	none	per-variable	competing	bivar.	bivar.	none	per-variable
GPU support	×	×	×	✓	✓	×	×	✓
Tested d (nested real-data)	35	∼10	N/A	6	2	2	98	98

Both apply to low dimensions. HACSurv [Liu et al., 2025b] fits hierarchical Archimedean copulas for competing risks where each observation reduces to one event time and the likelihood is a single first-order partial of the nested CDF. We handle the high-order mixed partials that arise with per-variable right-censoring in the survival sense, and discuss interval-censoring in Section 6.

- *Deep and probabilistic ML copulas.* Recent neural-copula work [Foomani et al., 2023, Liu et al., 2025a] learns flexible generators for bivariate dependent censoring and discrete-diffusion joint sampling; we are complementary, targeting exact nested likelihoods at moderate-to-high d with per-variable censoring. Vine copulas [Aas et al., 2009, Nagler and Vatter, 2023, Stöber et al., 2013] and hierarchical Kendall copulas [Brechmann, 2014] are specified through their densities and lack a closed-form CDF, so the censored likelihood of each observation requires a multivariate integral over its censored variables, whose cost grows exponentially with their number. METIC [Chen et al., 2025] handles this at small d via a stochastic-numerical scheme; to our knowledge, no vine implementation reaches per-variable censoring at d=53 (Appendices W and X).
- *Higher-order AD and differentiable probabilistic systems.* Our truncated-Taylor-series computation uses JAX's Taylor-mode AD primitive [Bettencourt et al., 2019], also formalized in the Weil-algebraic framework of Betancourt [2018]: order-d derivatives in a single forward pass on a truncated polynomial algebra (Section 3.2). Copulas.jl [Laverny and Jimenez, 2024] evaluates single-layer Archimedean copula densities by Taylor-mode AD; our work generalizes to arbitrary-depth nested trees. Lin [2024] extends AD to functional derivatives, complementary to our Taylor-mode AD. NumPyro [Phan et al., 2019] exposes first-order gradients, whereas nested copula densities require higher-order derivatives; Gen-AC [Ng et al., 2021] learns Laplace transforms with Monte Carlo variance that grows with derivative order.

Roadmap. Sections 2 to 4 formulate the problem, present the algorithm, and describe the implementation. Section 5 validates correctness in simulation, compares speed with R, and reports the real-data experiments; Section 6 discusses nesting validity, identifiability, and practical limitations.

2 Background and Notation

A *copula* $C\colon [0,1]^d \to [0,1]$ captures the dependence among d uniform random variables, independent of their marginal distributions [Sklar, 1959, Embrechts et al., 2002, Nelsen, 2006]. An *Archimedean copula* describes that dependence through a single *generator* $\psi\colon [0,\infty) \to [0,1]$—a continuous, strictly decreasing function with $\psi(0)$=1 that is d-monotone, equivalent to convex for d=2 and ensured by complete monotonicity for any d [McNeil and Nešlehová, 2009], producing the CDF:

$$C(\mathbf{u}) = \psi\Big(\sum_{j=1}^{d} \psi^{-1}(u_j)\Big), \qquad \mathbf{u} \in [0,1]^d. \tag{1}$$

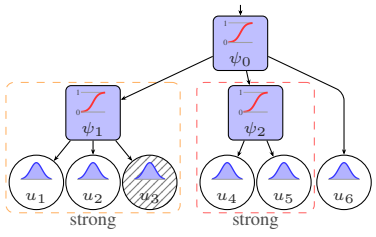


Figure 1: Nested Archimedean copula on six leaves. Each node has a generator that forms the copula CDF of its children; internal subtrees capture within-group dependence. Leaves are variables with marginal densities; the striped leaf u_3 is censored.

Nested Archimedean copulas [McNeil, 2008] relax the flat exchangeability of Equation (1) by arranging generators in a tree (Figure 1). Each node v has its own generator ψ_v and the CDF follows a recursive composition over its children c:

$$C_v(\mathbf{u}_v) = \psi_v\Big(\sum_{c \in \text{children}(v)} \psi_v^{-1}\big(C_c(\mathbf{u}_c)\big)\Big). \tag{2}$$

Equation (2) defines a valid copula when each composition $\psi_{\text{outer}}^{-1} \circ \psi_{\text{inner}}$ is a *Bernstein function*, i.e., one with a completely monotone derivative, which is a sufficient condition [McNeil, 2008]; Section 6 covers same-family closed-form ordering and the mixed-family sign-alternation diagnostic.

Likelihood with per-variable censoring. Fitting requires the density, i.e., the mixed partials of Equation (2). Under per-variable censoring, each variable of an observation can be censored separately, so the set of censored variables differs across observations. Each observation i has a *censoring mask* $\delta^{(i)} \in \{0,1\}^d$ with $\delta_j^{(i)} = 1$ if variable $j \in \{1,\dots,d\}$ is observed and 0 if it is censored; $|\delta| := \sum_j \delta_j$ counts the uncensored variables. Without loss of generality, we consider right-censoring in the survival sense where marginals are $u_j = S_j(t_j) := \Pr(T_j > t_j)$ and $C(\mathbf{u}) = \Pr(T_1 > t_1, \dots, T_d > t_d)$, with right-censored observations $T_j > c_j$ entering as $u_j = S_j(c_j)$. We assume that censoring is independent of the event times; under informative censoring, the censoring time may join the model as one more margin, following Li et al. [2023]. The likelihood is the order-$|\delta|$ mixed partial in the uncensored variables:

$$c_\delta(\mathbf{u}) = \frac{\partial^{|\delta|} C}{\prod_{j:\delta_j=1} \partial u_j}(\mathbf{u}), \qquad \ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log c_{\delta^{(i)}}(\mathbf{u}^{(i)};\, \boldsymbol{\theta}),$$

with sample size n, observation $\mathbf{u}^{(i)} \in [0,1]^d$, and $\boldsymbol{\theta}$ stacking all generator parameters. Two prior routes evaluate c_δ: hand-deriving a closed form per δ—infeasible at 2^d patterns—or nesting $|\delta|$ first-order AD calls at $O(2^{|\delta|})$ from intermediate trace blow-up [Betancourt, 2018]; neither scales past $d \approx 10$. R's `copula` package, the dominant alternative, uses hand-derived Stirling-number tables for five families [Hofert et al., 2012, Hofert and Pham, 2013], which avoids the exponential cost but covers just single-family two-level nesting and does not apply to censored likelihoods or provide analytic gradients.

The Hofert and Pham [2013] density. Hofert and Pham [2013] expressed the nested density c_δ as a tree-structured sum involving *partial Bell polynomials* $B_{n,k}$ of higher-order derivatives of each parent–child edge composition $h_{vc} = \psi_v^{-1} \circ \psi_c$ (writing $\psi^{(k)}$ for the k-th derivative throughout):

$$c_\delta(\mathbf{u}) = \left(\sum_{k=1}^{|\delta|} \beta_k^{(r)} \cdot \psi_r^{(k)}(t_r) \right) \cdot \prod_{\substack{\ell \in \text{leaves} \\ \delta_\ell = 1}} \left(\psi_{\text{par}(\ell)}^{-1}\right)'(u_\ell), \tag{3}$$

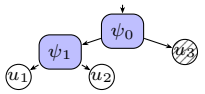
with r the root, $\text{par}(\ell)$ the parent of leaf ℓ, $t_v := \sum_{c \in \text{children}(v)} \psi_v^{-1}(C_c(\mathbf{u}_c))$ the inverse-generator argument at node v, and weights $\beta_k^{(r)}$ defined recursively in Section 3.2—Appendix A.1 gives the $B_{n,k}$ formula, and Appendix A.6 derives Equation (3) from Equation (2) on a three-variable tree. Evaluating these Bell polynomials for user-defined generators without hand-derived tables is the obstacle Section 3 resolves by powering Taylor coefficients.

Taylor-mode AD. Our algorithm needs univariate higher-order derivatives of each edge composition $h = \psi_v^{-1} \circ \psi_c$. *Taylor-mode* AD [Griewank and Walther, 2008], or *jet propagation*, returns the factorial-scaled derivative vector $\text{JET}(f, x, n) = \big(f(x), f'(x)/1!, \dots, f^{(n)}(x)/n!\big)$ in one forward pass at $O(n^2)$, versus $O(2^n)$ for n nested first-order AD calls. Our implementation builds on a JIT-friendly variant of JAX's `jet` [Radul et al., 2023]; Section 3 treats it as a black box.

3 Method

We walk through a concrete example exhibiting every algorithmic ingredient (Section 3.1), state the bottom-up recursion (Section 3.2; full Bell-polynomial derivation in Appendices A.1 and A.3), give the $O(d^3)$ complexity, and treat the per-variable censoring mechanism.

3.1 Worked example: d=3 nested copula with one censored leaf



Consider the tree on the right: a root ψ_0 with sector ψ_1 over uncensored leaves u_1, u_2 and a right-censored sibling u_3. The censoring mask $\delta = (1,1,0)$ gives an order-2 mixed partial $\partial^2 C / \partial u_1 \partial u_2$

at $\mathbf{u}$. Each leaf has an α-*vector*, a coefficient vector representing the leaf's contribution to its parent's density: $\alpha^{(u_1)} = \alpha^{(u_2)} = [0, 1]$ and $\alpha^{(u_3)} = [1]$, the Cauchy-product identity. Four operations propagate these vectors upward and yield the density.

1. *Cauchy product at the sector.* Combine the uncensored leaves: $\beta^{(\psi_1)} = \alpha^{(u_1)} * \alpha^{(u_2)} = [0, 0, 1]$, where $\beta_2^{(\psi_1)} = 1$ records the two differentiations ψ_1 contributes upstream.
2. *Bell transform across the edge* $\psi_0 \leftarrow \psi_1$. Convert $\beta^{(\psi_1)}$ into a child-wise α-vector at the root using higher-order derivatives of the edge composition $h = \psi_0^{-1} \circ \psi_1$ at $t_1 = \psi_1^{-1}(u_1) + \psi_1^{-1}(u_2)$: $\alpha_k^{(\psi_0,\psi_1)} = \sum_{j=k}^{2} \beta_j^{(\psi_1)} B_{j,k}(h'(t_1), h''(t_1), \ldots)$. Only $j = 2$ contributes, giving $\alpha_1^{(\psi_0,\psi_1)} = h''(t_1)$ and $\alpha_2^{(\psi_0,\psi_1)} = (h'(t_1))^2$.
3. *Censored sibling.* The leaf u_3 contributes $\alpha^{(u_3)} = [1]$, which leaves $\alpha^{(\psi_0,\psi_1)}$ unshifted under the Cauchy product at the root: $\beta^{(\psi_0)} = \alpha^{(\psi_0,\psi_1)} * [1] = [0, h''(t_1), (h'(t_1))^2]$.
4. *Density assembly.* The Hofert formula Equation (3) gives

$$c_\delta(\mathbf{u}) = \left[\psi_0'(t_r)\, h''(t_1) + \psi_0''(t_r)\, (h'(t_1))^2\right] \cdot (\psi_1^{-1})'(u_1) \cdot (\psi_1^{-1})'(u_2), \tag{4}$$

with $t_r = \psi_0^{-1}\big(C_1(u_1, u_2)\big) + \psi_0^{-1}(u_3)$ and C_1 the inner copula generated by ψ_1. The censored leaf u_3 enters t_r via its inverse-generator term but drops out of the leaf product since $\delta_3 = 0$.

The *Bell polynomial values* in Equation (4) — $h''(t_1)$ at k=1 and $(h'(t_1))^2$ at k=2—equal $n!/k!$ times the coefficients of ε^n in $p(\varepsilon)^k$ (here n=2, with 2!/1!=2 for k=1 and 2!/2!=1 for k=2), where $p(\varepsilon) = h'(t_1)\,\varepsilon + h''(t_1)\,\varepsilon^2/2 + \cdots$ (Appendix A.3). Taylor-mode AD applied to h at t_1 returns these coefficients of p in one forward pass; raising p to the k-th power by truncated convolution yields all required Bell polynomial values without combinatorial enumeration or per-family tables. The Cauchy product, the Bell transform, and the polynomial powering of Taylor coefficients are the entire algorithm.

Algorithm 1 Main algorithm.

Require: Copula tree of valid nesting at root r, observations $\mathbf{u}$
Ensure: Log-density $\log c(\mathbf{u})$

Compute $t_v = \sum_c \psi_v^{-1}(C_c)$ at every node v (bottom-up)
for each node v, from leaves to root **do**
  for each child c of v **do**
    if c is an uncensored leaf **then**
      $\alpha^{(c)} \leftarrow [0,\ 1]$ ▷ *one derivative, coefficient 1*
    else if c is a censored leaf **then**
      $\alpha^{(c)} \leftarrow [1]$ ▷ *identity under convolution*
    else
      $p \leftarrow \textsc{Jet}\big(\psi_v^{-1} \circ \psi_c,\ t_c,\ d_c\big)$ ▷ *Taylor coeffs of h_{vc}, computed via implicit equation solve in Appendix A.4*
      $q \leftarrow p$ ▷ *q accumulates p^k; starts at p^1*
      for $k = 1, \ldots, d_c$ **do**
        $\alpha_k^{(c)} \leftarrow (1/k!)\,\langle \tilde{\beta}^{(c)},\ q \rangle$ ▷ *reads off $B_{j,k}$*
        $q \leftarrow q * p$ ▷ *$q = p^{k+1}$*
  $\beta^{(v)} \leftarrow \alpha^{(c_1)} * \alpha^{(c_2)} * \cdots$ ▷ *convolve children*
$q_1, \ldots, q_d \leftarrow \textsc{Jet}(\psi_r, t_r, d)$ ▷ *root Taylor coeffs*
return $\log\Big(\big(\sum_k \beta_k^{(r)} \cdot k!\, q_k\big) \cdot \prod_{\ell:\, \delta_\ell = 1} (\psi_{\mathrm{par}(\ell)}^{-1})'(u_\ell)\Big)$

3.2 Algorithm

The general procedure repeats the worked example bottom-up at every internal node. Each child c of node v, where $m_v = |\mathrm{children}(v)|$, has an α-vector $\alpha^{(v,c)}$, and the β-*vector* $\beta^{(v)}$ collects the combined contribution of children of v; uncensored leaves initialise $\alpha = [0, 1]$ and censored leaves use $\alpha = [1]$, the Cauchy-product identity. Let δ_v be the count of uncensored leaves in the subtree at v. Two operations propagate these vectors upward, the *Bell transform* (Equation (5)) across each edge $v \leftarrow c$ and the *Cauchy product* (Equation (6)) combining siblings:

$$\forall k = 1, \ldots, |\delta_c|: \qquad \alpha_k^{(v,c)} = \sum_{j=k}^{|\delta_c|} \beta_j^{(c)}\ B_{j,k}\big(h'_{vc}(t_c), h''_{vc}(t_c), \ldots\big), \tag{5}$$

$$\forall k = 0, \ldots, |\delta_v|: \qquad \beta_k^{(v)} = \sum_{k_1 + \cdots + k_{m_v} = k}^{\forall k_i \geq 0} \alpha_{k_1}^{(v,c_1)} \cdots \alpha_{k_{m_v}}^{(v,c_{m_v})}. \tag{6}$$

The recursion follows Hofert and Pham [2013]; our contribution is evaluating the Bell polynomials $B_{j,k}$ in Equation (5) via polynomial powering of Taylor coefficients. With ε a formal indeterminate and $p_{vc}(\varepsilon) = \sum_{m \geq 1} h_{vc}^{(m)}(t_c)\, \varepsilon^m / m!$ the exponential generating function of the edge derivatives, a generating-function identity gives $B_{n,k} = (n!/k!) \cdot \mathrm{Coeff}\big(p_{vc}(\varepsilon)^k;\ \varepsilon^n\big)$, where $\mathrm{Coeff}(q;\ \varepsilon^n)$ extracts the coefficient of ε^n in the polynomial q. Applying Taylor-mode AD to h_{vc} at t_c returns the coefficients of p_{vc} directly; iterated truncated convolution $q \leftarrow q \cdot p_{vc}$ produces p_{vc}^k. Substituting back, Equation (5) reduces to a single inner product $\alpha_k^{(v,c)} = (1/k!) \sum_{j=k}^{|\delta_c|} \tilde{\beta}_j^{(c)}\, (q_k)_j$ with $\tilde{\beta}_j := j!\, \beta_j$ the rescaled β-vector and $(q_k)_j$ the ε^j coefficient of p_{vc}^k.

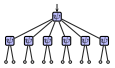
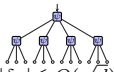
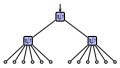
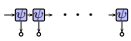
Algorithm 1 summarises the resulting bottom-up traversal. Every step is differentiable in the parameters, so first-order AD of the log-likelihood enables gradient-based optimisers. For numerical stability, the algorithm performs operations in the log-domain (Appendix B). Moreover, at Line 9, rather than feeding the explicit composition $\psi_v^{-1} \circ \psi_c$ to JET, the algorithm computes p_{vc} by solving the implicit equation $\psi_v(h_{vc}(t)) = \psi_c(t)$ for its Taylor coefficients to avoid the catastrophic cancellation that arises when $\psi_c(t)$ underflows before ψ_v^{-1} can recover it (Appendix A.4). In principle, the algorithm can also use arbitrary-precision arithmetic to avoid these tricks at the cost of efficiency.

Complexity Let $T(\mathcal{T})$ denote the total runtime of Algorithm 1 on nesting tree $\mathcal{T}$. Each non-root node v pays $O(|\delta_v|^3)$ for the polynomial powering its parent applies to it, which dominates the $O(|\delta_v|^2)$ Cauchy product at any internal v, giving

$$T(\mathcal{T}) \;=\; O\Big(d^2 + \textstyle\sum_{v \neq \text{root}} |\delta_v|^3\Big),$$

where d^2 comes from the order-d jet at the root. The largest subtree therefore dominates: total cost reaches the $O(d^2)$ root-jet cost whenever $\sum_{v \neq r} |\delta_v|^3 = O(d^2)$, a condition met for $\sqrt{d}$-*decomposed* trees—those where every non-root subtree satisfies $|\delta_v| = O(\sqrt{d})$ at constant tree depth—since per-level disjointness gives $\sum_{v \text{ at level}} |\delta_v|^3 = O(d^2)$. Any subtree exceeding $\sqrt{d}$ raises the total by its $|\delta_v|^3$ contribution. Table 2 lists the common cases, and Appendix A.7 derives the bound rigorously. Each edge's implicit-equation solve costs $O(|\delta_c|^3)$, matching the Bell powering it feeds, so it is absorbable into the total bound (Appendix A.4).

Table 2: Asymptotic cost by tree shape, governed by the largest internal-child subtree leaf count $|\delta_v|$. Any shape with $\max |\delta_v| \le O(\sqrt{d})$ at every level reaches the $O(d^2)$ root-jet cost. †Chains do not appear in real copula data to our knowledge. Derivation in Appendix A.7.

Constant sector size	Square-root scaling	Two large sectors	Chain (depth d)†
			
$\lvert\delta_v\rvert{=}O(1)$	$\lvert\delta_v\rvert{\le}\, O(\sqrt{d})$	$\lvert\delta_v\rvert{=}d/2$	$\lvert\delta_v\rvert{=}\Theta(d{-}\ell)$
$O(d^2)$	$O(d^2)$	$O(d^3)$	$O(d^4)$

In contrast, prior tools make weaker trade-offs: R's `copula` relies on hard-coded Stirling tables for five families at two nesting levels, and R's symbolic differentiation `D()` [Li et al., 2020, 2023] incurs a combinatorial complexity tractable only for $d \lesssim 6$. Appendix D discusses FFT and scan-based accelerations for the polynomial-powering step.

Handling per-variable censoring Censoring a leaf ℓ sets $\alpha^{(\ell)} = [1]$ instead of $[0, 1]$, which leaves the Cauchy product unchanged on its level and reduces the effective dimension $|\delta_v|$ at every ancestor. The reduction propagates automatically: shorter Cauchy product, lower Bell-polynomial degree, and reduced JET order; per-node cost scales as $O(|\delta_v|^3)$, so heavy censoring runs faster (Table 7).

4 Implementation

```python
@copula
class Clayton:
  theta: float
  def generator(self, t):
    return ((1 + t) **
            (-1.0 / self.theta))

@copula
class AMH:
  theta: float
  def generator(self, t):
    e = jnp.exp(-t)
    return ((1 - self.theta) * e /
            (1 - self.theta * e))
```

```python
def nested_amh_clayton(p, u):
  return  AMH(p.theta_outer)(
            Clayton(p.theta_inner)(
              marginal(Weibull(1.5, 1.0),
              obs=u[i, j],
              censored=((i, j) == (1, 3)))
            for j in range(5))
          for i in range(4))

p0 = Params(theta_outer=0.5, theta_inner=2.0)
obs = load_data()
m = compile_model(nested_amh_clayton, template=p0)
m.ll_fn(obs, m.flatten(p0))  # log density
jax.grad(m.ll_fn, argnums=1)(obs, m.flatten(p0))  # exact grad
```

Listing 1: A nested AMH/Clayton copula with 4 sectors of 5 Weibull-marginal leaves, per-variable censoring, and end-to-end differentiable inference.

ACOPULA is an open-source JAX [Bradbury et al., 2018] package.[1] The user supplies a generator class; the framework derives the inverse, all higher-order derivatives, the Bell-polynomial coefficients, the density values, and the parameter gradients automatically. Listing 1 shows two custom families and a 4-sector nested model with per-variable censoring fitted in fewer than twenty Python lines.

[1] Available at `https://github.com/thisiscam/acopula`.

The `@copula` decorator registers parameters and derives the generator inverse via symbolic inversion [Radul et al., 2023]; when symbolic inversion fails, ACOPULA falls back to bisection with implicit-function-theorem (IFT) gradients (Appendix E). ACOPULA compiles a single likelihood program that handles every per-observation censoring mask $\delta^{(i)}$ without recompilation (Appendix A.7). As is standard in AD-based frameworks, users may override any of the framework's automatic derivations with a hand-derived alternative — a custom generator inverse, a closed-form edge composition, or a per-family Taylor-coefficient routine in place of JET — for performance or numerical stability.

5 Experiments

We verify correctness on simulated data, characterise wall-clock scaling against R's `nacLL`, then evaluate three real-data benchmarks that each exercise a capability prior tools cannot deliver: per-variable censoring at d=53 on MIMIC-IV, nested likelihood at d=98 on S&P 500, and family-agnostic censored MLE on retinopathy. All runs share a single JAX code base in float64. The scaling sweep, the recovery study, R comparison, and S&P 500 and retinopathy fit run on a single AMD Threadripper 3960X CPU core in single-process mode for the apples-to-apples R comparison; the MIMIC-IV experiments run on an NVIDIA RTX 3090 Ti GPU (Table 27).

5.1 Correctness validation

We verify the implementation in three ways before turning to real data. The *parameter-recovery* experiment samples from a nested Clayton model with known true parameters, refits by maximum likelihood, and compares the estimate to the truth. Table 3 reports recovery with R=500 Monte Carlo replications, $d \in \{10, 20, 50\}$, and tree depth up to three: every parameter recovers to within 2.7% bias, and the empirical SE matches the observed-Fisher SE within 2.9%, confirming that Hessian-based inference on the Bell-polynomial likelihood is well-calibrated. We verify density agreement with R's `nacLL` to $<10^{-14}$ and with Mathematica's symbolic derivatives up to d=60 (Appendix J), and censored densities to $<10^{-14}$, including nested trees with censored leaves in subtrees of unequal size (Appendix N). AD gradients match central differences to $<10^{-8}$ in Appendix L; Appendix M.1 extends parameter recovery to Frank and Gumbel.

Table 3: Replicated parameter recovery on nested Clayton copulas (500 replications, 500 samples per replication; L denotes nesting depth, EmpSE the across-replication SE, HessSE the observed-Fisher SE). Bias stays under 2.7% of the true value; the empirical SE matches the observed-Fisher SE. Results suggest standard Hessian-based confidence intervals on our exact-likelihood fits are reliable.

d/L	Param	True	Mean	EmpSE	HessSE
10/2	θ_{outer}	1.50	1.521	0.069	0.070
	θ_{inner}	3.00	3.023	0.054	0.053
20/2	θ_{outer}	2.00	2.019	0.048	0.047
	θ_{inner}	4.00	4.023	0.048	0.047
50/3	θ_{outer}	1.00	1.016	0.052	0.050
	θ_{middle}	2.00	2.016	0.028	0.028
	θ_{inner}	4.00	4.023	0.032	0.030

5.2 Scaling: comparison with R `nacLL`

A wall-clock comparison against R's `nacLL` under identical single-thread CPU conditions at n=1 per evaluation shows ACOPULA growing polynomially in d while `nacLL` grows roughly exponentially in the number of sectors: by d=35 R is already ∼650× slower per density, and from d=40 its runs exceed our harness's 900-second cap without erroring (Figure 2 left); on the same recovery task, our exact gradients also beat R's frailty-sampled SMLE by ∼100× at comparable accuracy (Appendix R). ACOPULA extends across topologies that R cannot represent, since `nacLL` restricts to nesting depth two with same-family pairs; both 3-level fixed sector size K=5 leaves per inner node and balanced $\sqrt{d}$ reach the theoretical $O(d^2)$ asymptote at $d \in [6{,}400,\ 8{,}000]$ (Figure 2 right, Appendix P).

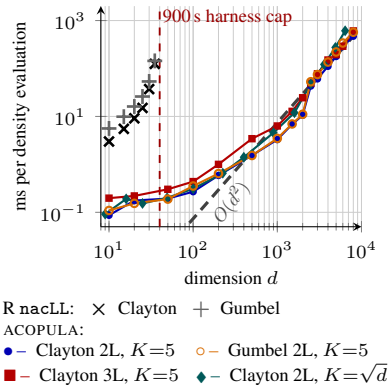


Figure 2: Wall-clock scaling. R only runs at 2-level K=5 and hits our 900 s harness cap from d=40. By d=35 R is ∼650× slower. ACOPULA extends across topologies R cannot represent and reaches $O(d^2)$ at $d \geq 6{,}400$.

5.3 MIMIC-IV: classical nested copulas at d=53

The MIMIC-IV ICU cohort exercises per-variable censoring at a dimension no prior nested-copula tool reaches. We

fit two-level nested Archimedean copulas on $85{,}229$ ICU admissions from MIMIC-IV [Johnson et al., 2023] at $d{=}53$ lab tests; the unit of observation is one ICU admission (`hadm_id`). Each lab variable records the time for its value to become abnormal since ICU entry, right-censored at ICU discharge, i.e. at the ICU length of stay; per-variable censoring rates span from 11.8% for red blood cells to 95.8% for specific gravity, and every admission censors a different subset, producing thousands of distinct masks (Appendix S.1 gives the cohort construction, the 80/20 admission-level split at seed 42, and the held-out protocol). We group the labs into seven clinical panels—CBC, CMP, differential, liver, coagulation, ABG, and Other—and follow the semiparametric two-stage procedure [Genest et al., 1995, Joe and Xu, 1996, Li et al., 2020, 2023]: Kaplan–Meier marginals at stage 1, joint copula by L-BFGS-B with exact gradients at stage 2. With the Kaplan–Meier marginals held fixed, each classical fit estimates eight association parameters, one θ per panel plus one outer θ (the $|\theta|$ column of Table 4). The held-out NLL on the $17{,}046$ test admissions checks the fit of the two-stage estimate; it is not a claim that nested copulas predict better than vines or other model classes.

Clayton/Clayton attains the best held-out NLL among classical same-family pairs, and it does so at every one of five re-randomised splits (Table 4, top). The three gaps to Frank/Frank, AMH/Clayton, and Gumbel/Gumbel run from 0.339 to 0.936 nats per admission, and all three clear both a Holm correction [Holm, 1979] for the three comparisons and a simultaneous band over those three at once (Appendix S.1.2), so the classical ordering is decisive. Frank/Frank's symmetric tail beats Gumbel's upper-tail-only dependence. The fitted within-panel Kendall's τ is 0.68 for the differential panel, 0.54 for coagulation, and 0.54 for ABG. The outer level ties the seven panels at $\tau_{\text{outer}}{=}0.13$.[2] The AMH/Clayton case exercises cross-family nesting using Hofert [2010]'s closed-form nesting validity constraint.

Table 4: MIMIC-IV held-out evaluation on the published seed-42 split. Top: classical families (Section 5.3); ‡ marks the cross-family nesting. Bottom: nested neural Archimedean generators (Section 5.4). The column Δ gives the paired test-NLL gap to the pre-specified reference Clayton/Clayton with a bootstrap interval, and $\approx$ marks the intervals that cover zero (Appendix S.1.2).

		$-\bar{\ell}\downarrow$		gap to Clayton/Clayton $\downarrow$	
Family	$\lvert\theta\rvert$	train	test	Δ [95% CI]	tie
Frank/Frank	8	7.399	7.442	0.354 [0.329, 0.379]	
Gumbel/Gumbel	8	7.985	8.042	0.954 [0.906, 1.002]	
Clayton/Clayton	8	7.069	7.088	reference	
AMH/Clayton‡	8	7.441	7.457	0.369 [0.348, 0.390]	
ACNet$_{(4,4)/\text{I4}}$	70	7.058	7.091	0.004 [−0.006, 0.014]	$\approx$
ACNet$_{(8,8)/\text{I4}}$	130	7.058	7.091	0.004 [−0.006, 0.014]	$\approx$
ACNet$_{(4,4,4)/\text{I4}}$	90	7.058	7.092	0.004 [−0.006, 0.013]	$\approx$
Gen-AC$_{1.1\text{k}}$	1,142	7.228	7.248	0.160 [0.145, 0.176]	
Gen-AC-Fourier$_{260}$	260	7.141	7.165	0.077 [0.065, 0.089]	
Gen-AC-Fourier-$\int_{260}$	260	7.069	7.096	0.008 [−0.003, 0.018]	$\approx$
Gen-AC-Fourier$_{3.6\text{k}}$	3,622	7.084	7.102	0.014 [0.003, 0.025]	

5.4 Capability demonstrations on nested neural Archimedean copulas

Our framework encompasses any Archimedean generator, including neural generators [Ling et al., 2020, Ng et al., 2021, Liu et al., 2025b]. Two experiments exercise this capability: we fit ACNet and Gen-AC on MIMIC-IV, and we exactly reproduce HACSurv's pipeline.

Generators. *ACNet* [Ling et al., 2020] parameterises ψ as a softmax-convex cascade of positive exponentials, enumerating exponentially many mixture atoms through polynomially many parameters (Appendix I). *Gen-AC* [Ng et al., 2021] defines ψ as a Laplace transform of an MLP-pushforward frailty, $\psi(t) = \mathbb{E}_U[\exp(-\exp(\text{MLP}(U))\, t)]$ with $U \sim \text{Uniform}(0,1)$; HACSurv [Liu et al., 2025b] reuses this generator under a competing-risks loss. Two-level trees compose each outer with an ACNet-style Bernstein inner per panel, preserving complete monotonicity (Appendix G.1).

MIMIC-IV results. The neural generators split into those the cohort cannot separate from the best classical fit, Clayton/Clayton, and those it separates decisively, with train-to-test gaps below 0.04 nats throughout. The three ACNet configurations make up the first set. On the published split they sit 0.004 nats behind Clayton/Clayton; over five re-randomised splits they sit 0.005 nats ahead of it, and Clayton/Clayton leads at the published split alone, trailing at the other four. Neither direction survives a correction for the seven comparisons, so the ordering between Clayton/Clayton and the ACNet rows is unresolvable at this cohort size and no reading of the third decimal separates them (Appendix S.1.2). The width and depth ablation reads the same way: the three ACNet configurations differ by 0.0004 nats on the published split, which Table 4 rounds to one unit in the third decimal, against a paired standard error of 0.005, so varying either changes nothing the data

[2] These values describe the fit under the assumption that censoring is independent of the lab event times (Appendix S.1).

can resolve. Gen-AC at its HACSurv-sized $1{,}142$-parameter configuration heads the second set, trailing at 7.248 despite having $16\times$ ACNet's parameters, and its gap to Clayton/Clayton, 0.160 nats on the published split and 0.142 across the five, clears every correction we apply on both axes; the 260-parameter Gen-AC-Fourier row behaves the same way at 0.077 and 0.089. We isolate the cause as the smoothness prior that a small MLP imposes (Appendix H). With Fourier-feature [Tancik et al., 2020] input $u \mapsto [\cos(2\pi k u), \sin(2\pi k u)]_{k=1}^{K}$, Gen-AC-Fourier recovers most of that gap to 7.102 at 3.6k parameters. That row still separates from Clayton/Clayton inside each of the five splits taken singly, but its gap moves by a factor of six between them, so the across-split interval covers zero (Appendix S.1.2). The Gen-AC-Fourier-$\int_{260}$ row integrates over the frailty distribution by numerical integration instead of the Monte Carlo approximation used by Ng et al. [2021], eliminating the estimation bias (Section 6 and Appendix R).

Framingham reproduction. HACSurv [Liu et al., 2025b] restricts to competing risks with one observed event time, where the likelihood is a first-order partial of the CDF, a strict subset of the mixed partials that our framework handles. Table 5 shows that we exactly reproduce their results on the Framingham dataset.

Table 5: Our port matches Liu et al. [2025b, Table 7]; entries $\times 10^2$.

	Risk 1		Risk 2	
Method	C-index	IBS	C-index	IBS
HACSurv	72.0 ± 1.5	7.6 ± 0.2	77.4 ± 0.8	9.4 ± 0.4
ACOPULA	71.9 ± 1.9	7.9 ± 0.4	77.7 ± 0.7	9.9 ± 0.6

5.5 S&P 500: nested likelihood at d=98

An 11-sector nested Gumbel copula with 12 parameters fits 98 S&P 500 equities over $1{,}253$ trading days. A regular vine [Aas et al., 2009, Nagler and Vatter, 2023] with $1{,}109$ parameters wins out of sample by 14.3 nats per test day, so the experiment demonstrates a capability—an exact nested Archimedean likelihood and its exact gradients at d=98, a dimension no prior nested-copula tool reaches—and not a better fit than a vine (Appendix U).

5.6 Retinopathy: family-agnostic censored MLE

The retinopathy paired-eye study [Huster et al., 1989] is a deliberately small benchmark with d=2, n=197, and 60.7% censoring. We use it to demonstrate the family-agnostic API of our framework. We use the same driver code to fit ten Archimedean generators, including five generators that no existing tool supports. All ten converge.

We compare every family against Frank, the reference we pre-specify from training likelihood alone, and once we correct for the nine remaining comparisons only Nelsen9† separates from it, so four of the five generators with no prior implementation stay statistically indistinguishable from the classical ones. Eight of the ten families sit within 0.350 nats of one another on the held-out patients, with Nelsen12† a further 0.823 nats ahead (Table 6). The ordering inside that band is not informative. Under repeated 5-fold cross-validation over all 197 patients, we score every family on the patients its own fit never saw, and none separates from Frank once we correct for the nine comparisons; the best mean held-out likelihood then belongs to AMH instead of to the seed-42 leader Nelsen12† (Appendix S.2.2). On the two families CopulaCenR [Sun and Ding, 2020] supports, our estimates agree to 0.6% while CopulaCenR's Frank optimiser fails on this dataset (Appendix J).

Table 6: Retinopathy NLL totals across all 10 generators, on the training patients and on the 40 held-out patients (Appendix S.2); † marks the five generators with no prior censored implementation. The column Δ_{tot} gives the paired test-NLL gap to the pre-specified reference Frank with a bootstrap interval, and $\approx$ marks the intervals that cover zero (Appendix S.2.2).

	total $-\ell \downarrow$		gap to Frank, Δ<0 better	
Family	train	test	Δ_{tot} [95% CI]	tie
Nelsen12†	90.422	20.317	-0.823 $[-1.631, 0.066]$	$\approx$
Frank	89.423	21.140	reference	
InvGauss†	89.503	21.167	0.027 $[-0.216, 0.268]$	$\approx$
Gumbel	89.702	21.203	0.063 $[-1.302, 1.241]$	$\approx$
Nelsen13†	89.492	21.216	0.076 $[-0.158, 0.309]$	$\approx$
Nelsen17†	89.491	21.223	0.083 $[-0.147, 0.330]$	$\approx$
AMH	89.557	21.381	0.241 $[-0.164, 0.670]$	$\approx$
Clayton	89.683	21.420	0.280 $[-0.320, 0.898]$	$\approx$
Joe	90.059	21.490	0.350 $[-1.371, 1.869]$	$\approx$
Nelsen9†	92.827	24.586	3.446 $[\ 1.228, 5.698]$	

6 Discussions

We address concerns deferred from the main sections.

Nesting validity. Equation (2) requires that each edge composition be a Bernstein function. Every nested experiment in this paper uses a valid-by-construction parameterisation: the same-family delta parameterisation of McNeil [2008] for Clayton/Clayton, Frank/Frank, and Gumbel/Gumbel; Hofert [2010]'s closed-form shifted-softplus constraint for AMH/Clayton; and the Bernstein-composition construction of Appendix G.1 for the Gen-AC and ACNet rows. In each case, the nesting condition holds by construction without optimiser-side enforcement. The framework additionally returns a free sign-alternation check on the edge-composition Taylor coefficients per density evaluation, and Appendix G outlines a soft-penalty method for cross-family or custom generators that lack a closed-form valid-by-construction parameterisation.

Identifiability. Identifiability is a structural limitation every copula-based survival model inherits, including HACSurv [Liu et al., 2025b] and DCSurvival [Foomani et al., 2023]. The practitioner response is to make identifying assumptions through the choice of copula family [Zheng and Klein, 1995, Foomani et al., 2023]; the family becomes an explicit modelling choice rather than an implicit defect. Our family-agnostic inference makes that choice cheap: we enable the analyst to fit ten families with identical code and report sensitivity directly (Section 5.6).

Within-group exchangeability. A nested Archimedean copula makes the variables under any one node exchangeable, so a single parameter governs every pairwise association inside a group. The tree relaxes that assumption only across groups, and refining it further costs one node per distinction the modeller wants to draw. Pair-copula constructions place a separate bivariate copula on each edge and so represent within-group heterogeneity that no nested Archimedean tree of the same size can express, which is what the S&P 500 comparison of Appendix W.2 measures. This model class therefore suits data whose groups are close to exchangeable by construction (e.g., replicate assays, members of a cluster, instruments within a sector), with few and interpretable parameters.

Censoring scope. This work does not consider interval-censored data. In principle, ACOPULA supports this case by wrapping 2^m calls to Algorithm 1, where m is the number of interval-censored variables, for the inclusion–exclusion over rectangle corners of the CDF. The cost is exponential in m, and we are not aware of any sub-exponential algorithm to compute this CDF exactly.

Informative censoring. The MIMIC-IV fit assumes that ICU discharge censors each admission independently of its lab event times. The assumption is strong for this cohort, because illness severity can both lengthen the stay and hasten abnormal labs, as Appendix S.1 discusses. A violation biases the Kaplan–Meier marginals and the fitted association parameters, and leaves the correctness and scaling results intact. Modelling the censoring time as one more copula margin, as Li et al. [2023] do, lifts the assumption, and Algorithm 1 evaluates the resulting likelihood. We leave the application of this approach to data under informative censoring to future work.

Stochastic density approximations. Two prior approaches replace exact copula density evaluation with Monte Carlo: Simulated MLE (SMLE) samples frailties at the joint-likelihood level [Hofert et al., 2013]; Gen-AC samples latents at the generator-derivative level [Ng et al., 2021]. Both yield consistent but biased estimators, with bias $O(1/N)$ from Jensen's inequality on $\log$ composed with the nonlinear density. Appendix R empirically quantifies both: 1. SMLE at $N{=}100$ frailty samples per evaluation biases the inner Clayton parameter estimate by 0.36; 2. Gen-AC-Fourier at $N{=}32$ frailty samples per evaluation biases the NLL by 0.03–0.07 nats per observation. Our work enables exact density at high dimensions without finite-sample approximation bias.

Compile cost and memory. JAX compilation takes 5–30 s for an S&P-style $d{=}98$ nested likelihood; MIMIC-IV hessian `val+grad` at $d{=}53$ takes 40–59 s across families, broken down in Table 27. ACOPULA caches compiled programs to disk, so reruns avoid compilation overhead; peak memory stays at 1.4 GB at $d{=}98$ and 700 MB at $d{=}53$.

7 Conclusion

ACOPULA unifies nested Archimedean copula inference under one differentiable pipeline. Where prior tools offered only restricted slices—a few hand-coded families, two-level nesting, low-dimensional mixed-partial derivatives—ACOPULA lifts all three to arbitrary families, arbitrary nesting depth, and high-dimensional mixed-partial derivatives, in one open-source package, lowering the barrier to nested Archimedean copula inference for survival, credit-risk, and reliability practitioners. We hope ACOPULA enables interpretable dependence models for censored data with exact likelihoods.

Acknowledgments

We thank the anonymous reviewers and the area chair for their thorough reviews and constructive feedback. We also thank Frank Niu for reading the paper and for his helpful suggestions.

References

Kjersti Aas, Claudia Czado, Arnoldo Frigessi, and Henrik Bakken. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics*, 44(2), 2009.

Michael Betancourt. A geometric theory of higher-order automatic differentiation. *arXiv preprint arXiv:1812.11592*, 2018.

Jesse Bettencourt, Matthew James Johnson, and David Duvenaud. Taylor-mode automatic differentiation for higher-order derivatives in JAX. In *Program Transformations for ML Workshop at NeurIPS*, 2019.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3), 1986.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Eike Christian Brechmann. Hierarchical Kendall copulas: Properties and inference. *Canadian Journal of Statistics*, 42(1), 2014.

Xinyuan Chen, Yiwei Li, and Qian M. Zhou. Analysis of multivariate event times under informative censoring using vine copula. *arXiv preprint arXiv:2502.20608*, 2025.

Louis Comtet. *Advanced Combinatorics: The Art of Finite and Infinite Expansions*. D. Reidel, 1974.

Paul Embrechts, Alexander McNeil, and Daniel Straumann. Correlation and dependence in risk management: properties and pitfalls. In M. A. H. Dempster, editor, *Risk Management: Value at Risk and Beyond*, pages 176–223. Cambridge University Press, 2002.

Takeshi Emura and Yi-Hau Chen. *Analysis of Survival Data with Dependent Censoring: Copula-Based Approaches*. Springer, 2018.

Ali Hossein Gharari Foomani, Michael Cooper, Russell Greiner, and Rahul G. Krishnan. Copula-based deep survival models for dependent censoring. In *Conference on Uncertainty in Artificial Intelligence*, 2023.

Christian Genest, Kilani Ghoudi, and Louis-Paul Rivest. A semiparametric estimation procedure of dependence parameters in multivariate families of distributions. *Biometrika*, 82(3), 1995.

Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2008.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.

Marius Hofert. Sampling Archimedean copulas. *Computational Statistics & Data Analysis*, 52(12), 2008.

Marius Hofert. *Sampling Nested Archimedean Copulas: With Applications to CDO Pricing*. PhD thesis, Ulm University, 2010.

Marius Hofert and David Pham. Densities of nested Archimedean copulas. *Journal of Multivariate Analysis*, 118, 2013.

Marius Hofert, Martin Mächler, and Alexander J. McNeil. Likelihood inference for Archimedean copulas in high dimensions under known margins. *Journal of Multivariate Analysis*, 110, 2012.

Marius Hofert, Martin Mächler, and Alexander J. McNeil. Archimedean copulas in high dimensions: Estimators and numerical challenges motivated by financial applications. *Journal de la Société Française de Statistique*, 154(1), 2013.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2), 1979.

William J. Huster, Ron Brookmeyer, and Steven G. Self. Modelling paired survival data with covariates. *Biometrics*, 45(1), 1989.

Harry Joe and James J. Xu. The estimation method of inference functions for margins for multivariate models. Technical Report 166, Department of Statistics, University of British Columbia, 1996.

Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1), 2023.

Oskar Laverny and Santiago Jimenez. Copulas.jl: A fully distributions.jl-compliant copula package. *Journal of Open Source Software*, 9(94):6189, 2024.

Dongdong Li, X. Joan Hu, Mary L. McBride, and John J. Spinelli. Multiple event times in the presence of informative censoring: modeling and analysis by copulas. *Lifetime Data Analysis*, 26 (3), 2020.

Dongdong Li, X. Joan Hu, and Rui Wang. Evaluating association between two event times with observations subject to informative censoring. *Journal of the American Statistical Association*, 118(542), 2023.

Min Lin. Automatic functional differentiation in JAX. In *International Conference on Learning Representations*, 2024.

Chun Kai Ling, Fei Fang, and J. Zico Kolter. Deep Archimedean copulas. In *Advances in Neural Information Processing Systems*, 2020.

Anji Liu, Oliver Broadrick, Mathias Niepert, and Guy Van den Broeck. Discrete copula diffusion. In *International Conference on Learning Representations*, 2025a.

Xin Liu, Weijia Zhang, and Min-Ling Zhang. HACSurv: A hierarchical copula-based approach for survival analysis with dependent competing risks. In *International Conference on Artificial Intelligence and Statistics*, 2025b.

Alexander J. McNeil. Sampling nested Archimedean copulas. *Journal of Statistical Computation and Simulation*, 78(6), 2008.

Alexander J. McNeil and Johanna Nešlehová. Multivariate Archimedean copulas, d-monotone functions and ℓ_1-norm symmetric distributions. *The Annals of Statistics*, 37(5B), 2009.

Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine Learning*, 52(3), 2003.

Thomas Nagler and Thibault Vatter. *rvinecopulib: High Performance Algorithms for Vine Copula Modeling*, 2023. URL `https://CRAN.R-project.org/package=rvinecopulib`. R package, CRAN.

Roger B. Nelsen. *An Introduction to Copulas*. Springer, 2006.

Yuting Ng, Ali Hasan, Khalil Elkhalil, and Vahid Tarokh. Generative Archimedean copulas. In *Uncertainty in Artificial Intelligence*, 2021.

Ostap Okhrin and Alexander Ristig. Hierarchical Archimedean copulae: The HAC package. *Journal of Statistical Software*, 58(4), 2014.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv preprint arXiv:1912.11554*, 2019.

Alexey Radul, Adam Paszke, Roy Frostig, Matthew Johnson, and Dougal Maclaurin. You only linearize once: Tangents transpose to gradients. *Proceedings of the ACM on Programming Languages*, 7(POPL), 2023.

Mohsen Rezapour. On the construction of nested Archimedean copulas for d-monotone generators. *Statistics & Probability Letters*, 101, 2015.

Joseph P. Romano and Michael Wolf. Exact and approximate stepdown methods for multiple hypothesis testing. *Journal of the American Statistical Association*, 100(469), 2005.

Abe Sklar. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8, 1959.

Jakob Stöber, Harry Joe, and Claudia Czado. Simplified pair copula constructions—limitations and extensions. *Journal of Multivariate Analysis*, 119, 2013.

Tao Sun and Ying Ding. CopulaCenR: Copula based regression models for bivariate censored data in R. *The R Journal*, 12(1), 2020.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, 2020.

Waloddi Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18(3), 1951.

Ming Zheng and John P. Klein. Estimates of marginal survival for dependent competing risks based on an assumed copula. *Biometrika*, 82(1), 1995.

Appendix Contents

A Density via Generator Derivatives: Full Derivation **15**

A.1 Partial Bell Polynomials 15

A.2 The Bottom-Up Recursion 15

A.3 Computing Bell Polynomials via Polynomial Powering 16

A.4 Edge-Taylor coefficients via implicit differentiation 16

A.5 Worked Example: 2×2 Nested Copula 17

A.6 The Three-Variable Density from the Nested CDF 18

A.7 Complexity Analysis 19

B Numerical Stability of Taylor-Mode Derivatives **20**

C Log-Space Taylor-Mode Differentiation **23**

C.1 Why naive float64 fails for Frank-type generators 23

C.2 Architecture: $(\text{sign}, \log|c|)$ throughout 23

C.3 Crossing the bell boundary without leaving log-space 23

C.4 Validation 23

C.5 Cost and configurability 24

D Polynomial Multiplication Alternatives **24**

E Symbolic Inversion and Numeric Fallback **25**

F Rosenblatt Sampling via Generator Derivatives **25**

G Validity of Nested Constructions **26**
G.1 Validity of the nested Gen-AC/ACNet construction 28

H Gen-AC MLP-pushforward generator with Fourier features **28**

I Deep ACNet Generator **29**

J Detailed R Comparison **29**

K Relationship to Copulas.jl **31**

L AD Gradient Verification **32**

M High-Dimensional Parameter Recovery **32**
M.1 Frank and Gumbel Recovery 32

N Censoring Validation **34**

O Supported Archimedean Families **35**

P High-Dimensional Nested Scaling **35**

Q Memory and Compilation Benchmarks **36**

R Exact MLE vs. Simulated Maximum Likelihood **37**

S Predictive Validation **38**
S.1 MIMIC-IV 38
S.2 Retinopathy 44

T Head-to-Head with HACSurv on Framingham **47**

U S&P 500: Nested Likelihood at d=98 **48**

V S&P 500 Sector-Level Tail Dependence **48**

W Vine Copula Comparison **49**
W.1 In-sample fit 49
W.2 Held-out evaluation 50
W.3 Censoring support and computational pattern 52

X Hierarchical Kendall Copulas **52**

A Density via Generator Derivatives: Full Derivation

This appendix establishes that the Bell polynomial recursion of Section 3.2 computes the exact nested-Archimedean density and admits a polynomial-powering implementation whose cost matches the bounds we cite in the body. We give the partial Bell polynomial background, the bottom-up recursion at each internal node, the polynomial-powering reformulation of the Bell-transform step, a worked 2×2 example, a derivation of the three-variable density of Section 3.1 directly from the nested CDF, and a complexity analysis with empirical scaling.

A.1 Partial Bell Polynomials

For integers $n \geq k \geq 1$, the partial Bell polynomial $B_{n,k}(x_1, x_2, \ldots, x_{n-k+1})$ counts all partitions of $\{1, \ldots, n\}$ into exactly k non-empty blocks, weighted by the x_m values. These polynomials arise in Faà di Bruno's formula for the higher-order chain rule:

$$\frac{d^n}{dt^n}\big[f(g(t))\big] = \sum_{k=1}^{n} f^{(k)}(g(t))\; B_{n,k}\big(g'(t),\, g''(t),\, \ldots,\, g^{(n-k+1)}(t)\big).$$

They satisfy the exponential generating function identity [Comtet, 1974, §3.3]

$$\sum_{n \geq k} B_{n,k}(x_1, x_2, \ldots)\, \frac{t^n}{n!} = \frac{1}{k!} \left(\sum_{m \geq 1} \frac{x_m}{m!}\, t^m \right)^k, \tag{7}$$

which provides the bridge to the polynomial-powering algorithm. For quick reference, the following table lists the values for dimensions up to $d = 4$:

n	k	$B_{n,k}$
1	1	x_1
2	1	x_2
2	2	x_1^2
3	1	x_3
3	2	$3\,x_1 x_2$
3	3	x_1^3

A.2 The Bottom-Up Recursion

At each internal node v with children $c_1, \ldots, c_m$, the subtree coefficients $\beta_k^{(v)}$ follow from three stages.

Leaf base case. An uncensored leaf ℓ contributes $\alpha_1^{(v,\ell)} = 1$. A censored leaf contributes $\alpha_0^{(v,\ell)} = 1$, which acts as the identity in the subsequent Cauchy product.

Bell polynomial step. For internal child c with already-computed subtree coefficients $\beta_j^{(c)}$, define the edge composition $h_{vc}(t) = \psi_v^{-1}(\psi_c(t))$. The per-child coefficients at the parent level are:

$$\alpha_k^{(v,c)} = \sum_{j=k}^{d_c} \beta_j^{(c)}\; B_{j,k}(h'_{vc}(t_c(\mathbf{u})),\; h''_{vc}(t_c(\mathbf{u})),\; \ldots), \qquad k = 1, \ldots, d_c. \tag{8}$$

Cauchy product. The per-child α-polynomials combine via polynomial multiplication:

$$\beta_k^{(v)} = \sum_{\substack{k_1 + \cdots + k_m = k \\ k_i \geq 0}} \prod_{i=1}^{m} \alpha_{k_i}^{(v,c_i)}, \qquad k = 0, \ldots, d_v.$$

Root assembly. At the root r, evaluate the root generator's Taylor expansion at t_r up to order d via a single jet call, then form $\sum_{k=1}^{d} \beta_k^{(r)} \psi_r^{(k)}(t_r)$.

A.3 Computing Bell Polynomials via Polynomial Powering

We derive the polynomial-powering algorithm that replaces explicit Bell polynomial evaluation with iterated polynomial convolution.

Step 1. Let $x_m = h_{vc}^{(m)}(t_c(\mathbf{u}))$ and define $p(t) = \sum_{m\geq 1}(x_m/m!)\, t^m$.

Step 2. The coefficient of t^n in $p(t)^k$ collects all ways to choose k terms from copies of p with exponents summing to n:

$$[t^n]\, p(t)^k = \sum_{\substack{m_1+\cdots+m_k=n \\ m_i\geq 1}} \frac{x_{m_1}}{m_1!}\cdots\frac{x_{m_k}}{m_k!}.$$

Step 3. The generating function identity Equation (7) gives $B_{n,k} = (n!/k!)\,[t^n]\, p(t)^k$.

Step 4. Substituting into Equation (8):

$$\alpha_k = \sum_j \beta_j \cdot \frac{j!}{k!}\cdot [t^j]\, p(t)^k = \frac{1}{k!}\sum_j \underbrace{j!\,\beta_j}_{\tilde{\beta}_j}\cdot \underbrace{[t^j]\, p(t)^k}_{q_k[j]} = \frac{1}{k!}\,\langle \tilde{\beta},\ q_k\rangle.$$

Step 5. Taylor-mode AD returns $h^{(m)}(t_c(\mathbf{u}))/m!$, which equals the coefficient of t^m in $p(t)$. The jet output *is* the coefficient vector of p.

Step 6. Build $q_k = p^k$ iteratively via truncated polynomial convolution, implemented as `lax.scan` + `lax.conv_general_dilated`:

$$q \leftarrow p; \qquad \text{for } k = 1,\ldots,d_c:\ \ \alpha_k = \frac{\langle \tilde{\beta},\, q\rangle}{k!},\ \ q \leftarrow \mathrm{conv}(q,p)\big|_{\leq d_c}.$$

Each convolution of two length-d_c polynomials costs $O(d_c^2)$, iterated d_c times, giving $O(d_c^3)$ per child.

A.4 Edge-Taylor coefficients via implicit differentiation

The polynomial-powering recipe of Appendix A.3 consumes the edge Taylor coefficients $h_{vc}^{(m)}(t_c)/m!$. The naive realisation applies JET directly to the explicit composition $h_{vc} = \psi_v^{-1}\circ\psi_c$, propagating two univariate jets through the chain rule (Faà di Bruno) at $O(d_c^2)$. The framework does not take this path by default: it is unstable for many generator pairs — Frank at $d \geq 90$, Nelsen9 at moderate θ — because $\psi_c(t)$ underflows to zero in float64 before ψ_v^{-1} can recover it.

The framework instead recovers p_{vc} without ever evaluating $\psi_v^{-1}\circ\psi_c$ as a single chain: the implicit solve reads only the separate jets of ψ_v at h_0 and of ψ_c at t_c, both of which lie in the well-conditioned interior of their respective generator domains. This construction therefore avoids the composed-chain underflow that defeats the naive path. The same mechanism extends naturally to opaque generators (e.g. a neural Archimedean family [Ling et al., 2020, Liu et al., 2025b]) without an analytic inverse: the algorithm uses ψ_v^{-1} only at one point to compute h_0, with all higher-order Taylor information recovered algebraically from ψ_v's forward jet. We start from the implicit equation

$$\psi_v\big(h_{vc}(t)\big) \ = \ \psi_c(t). \tag{9}$$

Let $h_0 := \psi_v^{-1}(\psi_c(t_c))$ and write $h_{vc}(t_c+\varepsilon) = h_0 + Q(\varepsilon)$ with $Q(\varepsilon) = \sum_{m=1}^{d_c} q_m\,\varepsilon^m$. Denote the Taylor coefficients of the two generators by $a_m := \psi_v^{(m)}(h_0)/m!$ and $b_m := \psi_c^{(m)}(t_c)/m!$. Equation (9) then expands as

$$\sum_{m\geq 1} a_m\, Q(\varepsilon)^m \ = \ \sum_{m\geq 1} b_m\,\varepsilon^m, \tag{10}$$

where the constant terms cancel because $\psi_v(h_0) = \psi_c(t_c)$ by construction. Matching the coefficient of ε^k yields a triangular system in $q_1,\ldots,q_{d_c}$:

$$a_1\, q_k \ + \ \underbrace{\sum_{m\geq 2} a_m\,[\varepsilon^k]\, Q(\varepsilon)^m}_{\text{depends only on } q_1,\ldots,q_{k-1}} \ = \ b_k.$$

Algorithm 2 Implicit Taylor solve for p_{vc} when the user has registered no closed-form edge composition. All steps are JIT-compiled and differentiable in the generator parameters.

Require: $a = (a_1, \ldots, a_{d_c})$ and $b = (b_1, \ldots, b_{d_c})$, the rescaled Taylor coefficients of ψ_v at h_0 and of ψ_c at t_c
Ensure: $q = (q_1, \ldots, q_{d_c})$ satisfying Equation (10)
$q_1 \leftarrow b_1 / a_1$
$C[m, j] \leftarrow 0$ for all m, j; $\quad C[m, m+2] \leftarrow q_1^{m+2}$ for $m = 0, \ldots, d_c - 2$ ▷ *Initialise C from $Q^{(0)} = q_1\,\varepsilon$*
for $k = 2, \ldots, d_c$ **do**
  correction $\leftarrow \sum_{m=0}^{d_c-2} a_{m+2} \cdot C[m, k]$
  $q_k \leftarrow (b_k - \text{correction}) / a_1$
  Append q_k to Q at slot k
  for $m = 0, 1, \ldots, d_c - 2$ **do**
    $C[m, k+1] \leftarrow \sum_{i=1}^{k} q_i \cdot \mathrm{prev}_m[k+1-i]$, with prev_m as in Equation (11).

Each $[\varepsilon^k]\, Q^m$ involves q_j only for $j < k$, so we solve for q_k in increasing k once we can read the correction term.

Algorithm. The framework maintains the table $C[m, j] := [\varepsilon^j]\, Q(\varepsilon)^{m+2}$ as Q's coefficients fill in. At outer step k, the algorithm has finalised Q at slots $1, \ldots, k$, so columns $j \le k$ of C are final and only column $j = k + 1$ needs updating. The single-column recurrence is

$$C[m, k+1] \;=\; \sum_{i=1}^{k} q_i \cdot \mathrm{prev}_m[k+1-i], \qquad \mathrm{prev}_m \;=\; \begin{cases} Q & m = 0, \\ C[m-1,\, \cdot] & m \ge 1. \end{cases} \tag{11}$$

The reads $C[m-1, k+1-i]$ at step k touch columns $\le k$ — already final from prior steps — so all rows m in the $j = k + 1$ column update in parallel. Algorithm 2 writes this out.

Complexity. The correction read at step k is an $O(d_c)$ inner product. The column update for slot $k + 1$ does $d_c - 1$ length-k dot products, costing $O(d_c \cdot k)$. Summing over $k = 1, \ldots, d_c - 1$ gives $O(d_c^3)$ per implicit-solve call, matching the per-child polynomial-powering cost in Appendix A.3. Naively rebuilding the full $Q^2, \ldots, Q^{d_c}$ ladder at every outer step would inflate this to $O(d_c^4)$; Algorithm 2 keeps the ladder amortised across k by writing one column per step. At balanced $\sqrt{d}$ topology the per-edge cost is dominated by the $O(d_c^3)$ Bell powering of Appendix A.3, so the headline $T(\mathcal{T}) = O(d^2)$ bound of Equation (14) is unaffected by the implicit solve.

Closed-form override. Same-family pairs (Clayton, Gumbel, Joe, Frank, Nelsen9 with $\theta_v < \theta_c$) admit an algebraic simplification of $h_{vc}(t)$ — for instance, Clayton/Clayton has $h(t) = (1+t)^{\theta_v/\theta_c} - 1$. Users register such pairs via `register_composition` to bypass the implicit solve and apply a single $O(d_c^2)$ jet through the registered form, saving a factor of d_c per edge. The framework ships built-in registrations for the same-family pairs above, so the classical single-family experiments (S&P 500, MIMIC-IV with classical generators, retinopathy) take this fast path. The neural Archimedean rows in Table 4 (ACNet, Gen-AC) also bypass the implicit solve: their inner generator is constructed as $\psi_{\text{inner}}(t) = \psi_{\text{outer}}(b(t))$ with b a Bernstein function (Appendix G.1), so the edge composition is $h_{vc}(t) = b(t)$ in closed form. Cross-classical-family pairs without a registered closed form (e.g. AMH/Clayton) fall through to the default implicit solve.

A.5 Worked Example: 2×2 Nested Copula

We trace the polynomial-powering recursion through the smallest non-trivial nested copula—two sectors of two leaves—to expose how the per-child α-vectors and the Cauchy product at the root combine into the closed-form density. Consider root ψ_0 with two sectors, each governed by ψ_1 with two leaves. At each sector, $t_v = \psi_1^{-1}(u_{v1}) + \psi_1^{-1}(u_{v2})$, $C_v = \psi_1(t_v)$, and $t_r = \psi_0^{-1}(C_1) + \psi_0^{-1}(C_2)$.

Each sector has two leaf children, so $\beta_2^{(v)} = 1$. Define $h(t) = \psi_0^{-1}(\psi_1(t))$ and run jet at t_v to get p_v. Polynomial powering gives $\alpha_1^{(r,v)} = h''(t_v)$ and $\alpha_2^{(r,v)} = (h'(t_v))^2$. The Cauchy product

at the root yields:

$$\begin{aligned}\beta_2^{(r)} &= \alpha_1^{(r,1)}\,\alpha_1^{(r,2)},\\ \beta_3^{(r)} &= \alpha_1^{(r,1)}\,\alpha_2^{(r,2)} + \alpha_2^{(r,1)}\,\alpha_1^{(r,2)},\\ \beta_4^{(r)} &= \alpha_2^{(r,1)}\,\alpha_2^{(r,2)}.\end{aligned}$$

The density is $c(\mathbf{u}) = [\beta_2\psi_0''(t_r) + \beta_3\psi_0'''(t_r) + \beta_4\psi_0^{(4)}(t_r)] \cdot \prod_{i=1}^{4}(\psi_1^{-1})'(u_i)$.

A.6 The Three-Variable Density from the Nested CDF

The worked example of Section 3.1 takes Equation (3) as given. Here, we derive the same density directly from the nested CDF of Equation (2), so that every factor of Equation (4) traces back to the chain rule. We keep the tree of Section 3.1: a root ψ_0 with a sector ψ_1 over the uncensored leaves u_1, u_2 and a right-censored sibling u_3, so the mask is $\delta = (1, 1, 0)$ and the density is the mixed partial $c_\delta(\mathbf{u}) = \partial^2 C/\partial u_1 \partial u_2$. Unrolling Equation (2) once expresses the CDF as a composition of the two generators,

$$C(\mathbf{u}) = \psi_0\big(\psi_0^{-1}(C_1(u_1, u_2)) + \psi_0^{-1}(u_3)\big), \qquad C_1(u_1, u_2) = \psi_1(t_1), \tag{12}$$

with $t_1 = \psi_1^{-1}(u_1) + \psi_1^{-1}(u_2)$, and the edge composition $h = \psi_0^{-1} \circ \psi_1$ of Section 3.1 turns the root argument into $t_r = h(t_1) + \psi_0^{-1}(u_3)$.

The sector is a scalar bottleneck. The two uncensored leaves reach C only through the scalar t_1. With $g(t) := h(t) + \psi_0^{-1}(u_3)$, Equation (12) becomes $C(\mathbf{u}) = \psi_0(g(t_1))$, and the censored leaf u_3 sits inside g as an additive constant. Differentiating with respect to u_1 pulls out the factor $(\psi_1^{-1})'(u_1)$ by the chain rule, and differentiating the result with respect to u_2 pulls out $(\psi_1^{-1})'(u_2)$ while leaving the first factor untouched, because $(\psi_1^{-1})'(u_1)$ does not depend on u_2:

$$\frac{\partial^2 C}{\partial u_1\,\partial u_2}(\mathbf{u}) = \frac{d^2}{dt^2}\Big[\psi_0\big(g(t)\big)\Big]_{t=t_1} \cdot (\psi_1^{-1})'(u_1)\,(\psi_1^{-1})'(u_2). \tag{13}$$

The leaf product of Equation (3) is now in place, and the censored leaf is missing from it because we take no derivative in u_3.

Faà di Bruno at the root. What remains is the second derivative of the univariate composition $\psi_0 \circ g$, which Faà di Bruno's formula from Appendix A.1 expands at order $n = 2$:

$$\begin{aligned}\frac{d^2}{dt^2}\Big[\psi_0\big(g(t)\big)\Big] &= \psi_0'\big(g(t)\big)\,B_{2,1}\big(g'(t), g''(t)\big) + \psi_0''\big(g(t)\big)\,B_{2,2}\big(g'(t)\big)\\ &= \psi_0'\big(g(t)\big)\,h''(t) + \psi_0''\big(g(t)\big)\,\big(h'(t)\big)^2,\end{aligned}$$

where the second equality uses $B_{2,1} = x_2$ and $B_{2,2} = x_1^2$ from the table in Appendix A.1 together with $g' = h'$ and $g'' = h''$, because the additive constant $\psi_0^{-1}(u_3)$ vanishes under differentiation. Evaluating at $t = t_1$, where $g(t_1) = t_r$, and substituting into Equation (13) gives

$$c_\delta(\mathbf{u}) = \big[\psi_0'(t_r)\,h''(t_1) + \psi_0''(t_r)\,(h'(t_1))^2\big] \cdot (\psi_1^{-1})'(u_1)\,(\psi_1^{-1})'(u_2),$$

which is Equation (4).

Reading off the recursion. The derivation also accounts for each step of Section 3.1. The order $n = 2$ of the Faà di Bruno expansion equals the number of uncensored leaves below the sector, which the recursion records as $\beta^{(\psi_1)} = [0, 0, 1]$. The two Bell polynomials $B_{2,1}(h', h'') = h''(t_1)$ and $B_{2,2}(h') = (h'(t_1))^2$ are the Bell transform across the edge $\psi_0 \leftarrow \psi_1$, namely Equation (8) with $\beta^{(c)} = [0, 0, 1]$. The censored leaf contributes only the constant $\psi_0^{-1}(u_3)$ inside t_r, which is why it enters the Cauchy product at the root as the identity $[1]$. A sector with n uncensored leaves changes only the order of differentiation: the chain-rule step yields $\prod_j(\psi_1^{-1})'(u_j)$ times the n-th derivative of $\psi_0 \circ g$ at t_1, and Faà di Bruno expands that derivative into $\sum_{k=1}^{n}\psi_0^{(k)}(t_r)\,B_{n,k}\big(h'(t_1), \ldots, h^{(n-k+1)}(t_1)\big)$, the root sum of Equation (3).

A.7 Complexity Analysis

We bound the runtime of the Bell-polynomial recursion by summing the per-node powering cost over the tree, then specialize the bound to common shapes and confirm it against measured wall times.

General bound. At each non-root node v the dominant cost is polynomial powering: d_v iterations of $O(d_v^2)$ truncated convolution applied by v's parent, giving $O(d_v^3)$. The Cauchy product at v costs $O(m_v \cdot d_v \cdot d_{\max}(v))$ with $d_{\max}(v) = \max_c d_c$; because $m_v \le d_v$, this Cauchy cost never exceeds $O(d_v^3)$, so the powering term dominates. Computing the edge Taylor vector p_{vc} that the powering step consumes adds $O(d_v^3)$ via the default implicit solve of Algorithm 2, or $O(d_v^2)$ when a closed-form h_{vc} is registered (Appendix A.4); the $O(d_v^3)$ powering bound absorbs both. At the root r, the powering step is replaced by a single order-d jet call of cost $O(d^2)$. The total cost is therefore

$$T(\mathcal{T}) = O\left(d^2 + \sum_{v \neq r} d_v^3\right), \tag{14}$$

where $d_v \equiv |\delta_v|$ counts uncensored leaves in the subtree of v, and the bound holds uniformly across the three edge-Taylor implementations.

Height bound. At each depth level the subtrees are disjoint, so the d_v values at that level sum to at most d. Since each $d_v \le d$, we have $d_v^3 \le d_v \cdot d^2$, so the per-level contribution satisfies $\sum_{v \text{ at level } \ell} d_v^3 \le d^2 \sum_v d_v \le d^3$. Summing over the h non-root internal levels gives $\sum_{v \neq r} d_v^3 \le h \cdot d^3$, so $T(\mathcal{T}) = O(d^2 + h \cdot d^3) = O(h \cdot d^3)$. For the standard case of constant-depth copula trees with $h = 1$ or 2, the algorithm runs in $O(d^3)$ time.

$\sqrt{d}$-decomposed trees. When every non-root subtree satisfies $d_v \le c\sqrt{d}$ for a constant c, the per-level bound sharpens: $d_v^2 \le c^2 d$ gives $d_v^3 \le c^2\, d \cdot d_v$, so $\sum_{v \text{ at level } \ell} d_v^3 \le c^2\, d \sum_v d_v \le c^2\, d^2$. Summing over the h non-root internal levels yields $\sum_{v \neq r} d_v^3 \le h\, c^2\, d^2$, and the total cost collapses to $T(\mathcal{T}) = O(d^2)$ for constant depth—matching the $O(d^2)$ root-jet cost. More generally, the $T(\mathcal{T}) = O(d^2)$ regime holds whenever $\sum_{v \neq r} d_v^3 = O(d^2)$, for which $d_v = O(\sqrt{d})$ is a sufficient condition.

Two-level symmetric illustration. As a concrete instance, take a two-level tree with M sectors of $K = d/M$ leaves each. Substituting into Equation (14) gives $O(MK^3 + M^2K^2) = O(d^3/M^2 + d^2)$; the d^3/M^2 term dominates when M grows slower than $\sqrt{d}$, and the cost collapses to $O(d^2)$ when $M = \Omega(\sqrt{d})$, equivalently $K = O(\sqrt{d})$—the two-level instance of the general $|\delta_v| = O(\sqrt{d})$ condition above. Taking $M = 2$ sectors with $K = d/2$ leaves each gives $O(d^3)$, not $O(d^2)$.

Special cases. The following table summarises cost for common tree shapes.

Tree shape	**Cost**	**In terms of d**
Single-layer (no nesting)	one jet call	$O(d^2)$
Two-level (M sectors $\times$ K leaves)	$O(MK^3 + M^2K^2)$	$O(d^3/M^2 + d^2)$
Balanced b-ary, depth L	$O(d^3/(b^2-1))$	$O(d^3)$
Chain (depth $= d$)	$\sum_{k=1}^{d} k^3 = \Theta(d^4)$	$O(d^4)$

Comparison with alternative approaches. The following table contrasts our method with existing alternatives.

Method	**Cost**	**All families?**	**Max d**
Polynomial powering (ours)	$O(d^3)$	Yes	> 200
R `copula` (Stirling tables)	$O(d^2)$	5 built-in, ≤ 2 layers	$\sim$170
R `D()` symbolic-numeric [Li et al., 2020, 2023]	combinatorial	Yes (symbolic)	$\sim$6
Nested first-order AD [Betancourt, 2018]	$O(2^d)$	Yes	$\sim$8
Forward-over-reverse / mixed-mode AD	$O(d^3)$	Yes; $O(d)$ tangents per primitive	$\sim$30

Empirical scaling. The $O(d^3)$ worst case assumes the per-sector leaf count K grows with d. In practice, the application domain fixes the tree branching factor—$K = 5$ in our scaling benchmarks—so the number of sectors $M = d/K$ grows instead. The per-sector polynomial powering then costs $O(K^3)$, constant in d, and the total across all sectors is $M \cdot O(K^3) = O(d \cdot K^2) = O(d)$. The Cauchy product convolves M short kernels of length K with a d-length accumulator at $O(d \cdot K \cdot M) = O(d^2)$. Both terms carry small constants relative to the per-scan-iteration overhead in `lax.scan`, so the wall-time transition where the quadratic term begins to dominate sits at $d^* \approx 10^3$ at fixed K. Appendix P reports the empirical slopes that follow this prediction: sub-linear in the small-d regime, transient super-quadratic during the L1/L2 cache transition, and the clean $O(d^2)$ asymptote at $d \geq 4{,}000$ for both the fixed-K=5 and balanced $\sqrt{d}$ topologies.

Mask-aware static-shape JIT. ACOPULA pads each node's α-vector to the mask-independent length $|V_v| + 1$, where V_v is the full leaf set in the subtree of v. Each child of v then dispatches at runtime through `lax.switch` to one of three branches—censored leaf, uncensored leaf, or subtree—so a single JIT-compiled program covers every per-observation mask without recompilation.

Censoring reduces runtime. The cost bound in Equation (14) depends on d_v, the number of *uncensored* leaves in each subtree—not on the total number of leaves. Censoring a variable sets its α-vector to $[1]$, which shortens the polynomial powering and Cauchy-product steps at every ancestor. To quantify this effect, we time the d=50 nested Clayton configuration with 10 sectors of 5 leaves on n=100 observations at four censoring rates on a single AMD Threadripper 3960X CPU core. Table 7 reports the results: 75% censoring reduces the effective dimension from 50 to 13 and delivers a 13.6× speedup, in line with the $O(d_v^3)$ scaling of the per-vertex Bell recursion that dominates the cost at this dimension. The $O(|V_v|^2)$ Cauchy-product floor (Section 3.2) remains a strict lower bound but does not bind at d=50 on CPU.

Table 7: Wall-clock time for a nested Clayton (d=50, n=100) log-likelihood evaluation at varying censoring rates on a single AMD Threadripper 3960X CPU core. d_{eff} counts uncensored variables. Median of 20 JIT-compiled calls.

Censoring	d_{eff}	Time (ms)	Speedup
0%	50	32.0	1.0×
25%	38	11.8	2.7×
50%	25	6.4	5.0×
75%	13	2.4	13.6×

B Numerical Stability of Taylor-Mode Derivatives

A naive float64 implementation of the Bell pipeline underflows long before reaching the dimensions our experiments target, because high-order generator derivatives and the geometric powers p_1^j that the Bell transform multiplies in span hundreds of orders of magnitude. This appendix specifies the log-domain machinery that keeps every intermediate quantity representable and reports the stress tests that confirm its accuracy. Algorithm 3 states the log-domain version of Algorithm 1: per-index rescaling to neutralize geometric decay in the polynomial powering, signed logsumexp for the Bell-transform inner products and root assembly, and intermediate normalisation during the Cauchy product.

Generator derivative growth. Each family's derivative growth rate sets the precision ceiling that the log-domain machinery must accommodate. For Clayton, $\psi(t) = (1+t)^{-1/\theta}$, the Pochhammer identity $\psi^{(k)}(t) = (-1)^k (1/\theta)_k (1+t)^{-1/\theta - k}$ gives $|\psi^{(k)}(t)|/k! = O(k^{1/\theta - 1})$ via Stirling, so the jet-stored Taylor coefficients grow only polynomially in k for fixed θ and t. Joe and other double-exponential generators grow faster because the chain rule nests an exponential inside another. Empirically, every family matches Mathematica to relative error below 3×10^{-11} wherever `D[]` completes (Table 9), and Table 8 confirms finite log-density evaluation up to $d = 60$ for nine of the ten families and up to $d = 4$ for Nelsen9, whose d-completely monotone parameter range tightens with d.

Edge composition safeguards. The *edge composition* $h = \psi_{\text{outer}}^{-1} \circ \psi_{\text{inner}}$ that the nested density requires admits three qualitatively different evaluation paths across our ten families (Appendix A.4),

and Frank in particular needs a closed-form rewrite to stay within float64. Clayton, AMH, Joe, Gumbel, Nelsen9, and Nelsen12 provide closed-form inverses, so the composition introduces no numerical root-finding error. InverseGaussian relies on Oryx's symbolic inversion, which pattern-matches individual primitives to their algebraic inverses. Two families also expose a numerical hazard in the *nested* composition $\psi_v^{-1}(\psi_c(t))$ even with closed-form inverses: Frank at $d = 98$ overflows because intermediate log-sum-exp terms exceed float64, and Nelsen9 underflows because $\psi_c(t)$ collapses to zero before ψ_v^{-1} can recover it. Both families ship a registered closed-form edge composition (Appendix A.4) that algebraically cancels the offending exponentials, bypassing the default implicit-differentiation solve of Algorithm 2.

Precision at large sector size. At d=60 with sector size K=30, Appendix J reports a 3.5×10^{-5} discrepancy against R's `nacLL`. Mathematica's symbolic differentiation of the copula CDF confirms that our framework matches the ground truth to machine epsilon at d=60; the discrepancy originates in R's `nacLL`, not in our polynomial powering.

Generator-specific issues. Nelsen9's double-exponential structure $\psi(t) = \exp\big((1 - e^t)/\theta\big)$ underflows to zero in float64 whenever the exponent $(1 - e^t)/\theta$ drops below -745, which occurs at moderate t for small θ. We mitigate the underflow on three fronts. A per-family Taylor hook supplies the coefficients via the Touchard factorization $\psi^{(n)}(t) = \psi(t) \cdot T_n(-e^t/\theta)$, where Faà di Bruno on $\psi = \exp \circ g$ with $g^{(k)}(t) = -e^t/\theta$ for every $k \geq 1$ collapses the complete Bell polynomial at equal arguments to the Touchard polynomial T_n. The framework evaluates $\log|\psi^{(n)}(t)| = (1 - e^t)/\theta + \log|T_n(-e^t/\theta)|$, which keeps the underflow-prone exponential symbolic and confines the remaining float64 work to the polynomial T_n, whose direct sum loses at most n decimal digits to alternating-sign cancellation in the d-monotone-valid range. The hook bypasses jet on the closed-form ψ, whose JVP produces a NaN gradient when ψ underflows. The analytical $\psi^{-1}(u) = \log(1 - \theta \log u)$ replaces Oryx's symbolic inverse, which we found to be numerically unstable at large t. The closed-form nested composition $h(t) = t + \log\big(r + (1 - r)\, e^{-t}\big)$ with $r = \theta_v/\theta_c$ replaces the naive $\psi_v^{-1}(\psi_c(t))$ chain that would underflow at moderate t, and the framework registers it as the closed-form override for Nelsen9/Nelsen9 nestings in Appendix A.4.

Stress test across parameter regimes. Table 8 reports a systematic stress test that sweeps the (family, strength, dimension) grid. We map four target Kendall-tau strengths $\tau \in \{0.10,\ 0.40,\ 0.70,\ 0.90\}$ to each family's parameter via numerical inversion when no closed form exists, build a two-sector nested same-family copula at sector size $K \in \{2, 5, 10, 15, 20, 25, 30\}$ for total dimension d=$2K$, and evaluate the log-density at 100 random u-vectors per cell. A cell *passes* when every evaluation returns a finite value.

Table 8: Stability stress test: number of K values out of seven that pass at 100% finiteness on 100 random u-vectors per cell. Each row sweeps $K \in \{2, 5, 10, 15, 20, 25, 30\}$ for the listed Kendall-tau strength. AMH[‡] has bounded tau $\tau \leq 1/3$, so its τ=0.40, 0.70, 0.90 columns collapse to the same parameter; we still report each column to keep the grid uniform.

Family	τ=0.10	τ=0.40	τ=0.70	τ=0.90	total
Clayton	7/7	7/7	7/7	7/7	28/28
Gumbel	7/7	7/7	7/7	7/7	28/28
Frank	7/7	7/7	7/7	7/7	28/28
AMH[‡]	7/7	7/7	7/7	7/7	28/28
Joe	7/7	7/7	7/7	7/7	28/28
InverseGaussian[†]	7/7	7/7	7/7	7/7	28/28
Nelsen12[†]	7/7	7/7	7/7	7/7	28/28
Nelsen13[†]	7/7	7/7	7/7	7/7	28/28
Nelsen17[†]	7/7	7/7	7/7	7/7	28/28
Nelsen9[†§]	1/1	—	—	—	1/1
Total					**253/253**

Across all 253 cells in the sweep that fall within each family's d-completely-monotone parameter range—ten generator families, four strengths, seven dimensions—every cell returns finite log-densities

on all 100 random evaluation points, a 100% pass rate. Nelsen9 is a special case: the d-CM upper bound $\theta_{\max}(d)$ contracts as d grows ($\theta_{\max}(4)$=0.22, $\theta_{\max}(10)$=0.057, $\theta_{\max}(60)$=0.007), so only the τ=0.10, K=2 cell (d=4, $\theta \leq 0.2$) lies inside the valid range; the other 27 cells skip.

We attribute the broad success to the log-domain rescaling and signed-logsumexp arithmetic of Algorithm 3: per-index rescaling absorbs the geometric blow-up that the naive recursion would produce at high dimension, and signed logsumexp prevents catastrophic cancellation in the Bell-transform inner products. [‡]AMH's bounded tau $\tau \leq 1/3$ collapses the three high-strength columns to the same parameter, which is why all four columns repeat identically. [†]Marks the five extra families that existing nested-copula tools do not cover. [§]Nelsen9 is d-completely monotone only for $\theta \leq \theta_{\max}(d) = 1/|r_d|$, where r_d is the most negative root of the d-th Touchard polynomial $T_d(y) = \sum_j S(d,j)\, y^j$; the other 27 cells in this grid violate that bound.

Cross-family accuracy verification. Table 9 compares our framework against Mathematica's symbolic differentiation of the copula CDF across all ten families. Mathematica evaluates each cell at 100 digit working precision; our framework runs in IEEE-754 `float64` ($\approx$ 16 digits) and matches the Mathematica ground truth to relative error below 3×10^{-11} in every configuration through d=60 where `D[]` completes.

The framework exposes a per-family hook, `Copula.generator_taylor_coefficients(t, k)`, that returns the Taylor expansion $[\psi(t),\, \psi'(t)/1!,\, \ldots,\, \psi^{(k)}(t)/k!]$ directly, bypassing chain-rule cancellations Taylor-mode AD on the closed-form ψ would produce at high derivative order. The framework consults the hook everywhere it would otherwise call jet on ψ: at the root density assembly and at every nested composition $h(t) = \psi_{\text{outer}}^{-1} \circ \psi_{\text{inner}}(t)$. For AMH ($\psi(t) = (1-\theta)/(e^t - \theta)$), Taylor-mode AD on the closed form costs roughly two decimal digits per ten dimensions in `float64`; the override sums the same-sign Geometric$(1-\theta)$ frailty series $\psi^{(k)}(t) = (-1)^k \sum_{x \geq 1} x^k (1-\theta)\theta^{x-1} e^{-tx}$ in log-domain via signed-logsumexp, which has no cancellation and recovers machine precision ($\leq 2 \times 10^{-15}$) at every $K \leq 30$. Other completely monotone families (Frank, Joe, Gumbel, InverseGaussian) admit similar overrides via their respective frailty representations; we provide AMH because it was the only family whose chain-rule path materially hit the `float64` cancellation wall in the verified range.

Table 9: Numerical accuracy and symbolic-differentiation scaling on two-sector nested same-family copulas. "Max rel. err." is the largest relative error between our framework and Mathematica's symbolic `D[]` across the sector sizes where Mathematica completes; "max K" is the largest sector size Mathematica reaches within a 300 s timeout; "leaves" is the `LeafCount` of Mathematica's symbolic derivative at that K; "`D[]` time" is the wall time to compute it.

Family	$(\theta_{\text{outer}}, \theta_{\text{inner}})$	Max rel. err.	max K	leaves	`D[]` time
Clayton	$(2, 5)$	4.8×10^{-15}	30	5.1×10^{5}	2.7 s
Gumbel	$(2, 5)$	7.5×10^{-14}	15	7.7×10^{7}	155.7 s
Frank	$(2, 5)$	8.0×10^{-15}	15	7.2×10^{7}	125.4 s
AMH	$(0.3, 0.7)$	1.7×10^{-15}	30	2.4×10^{7}	43.4 s
Joe	$(2, 5)$	7.6×10^{-13}	15	1.1×10^{8}	178.0 s
InverseGaussian[†]	$(1, 3)$	7.6×10^{-16}	15	1.6×10^{8}	178.5 s
Nelsen9[†]	$(1/200, 1/100)$	3.0×10^{-11}	30	1.9×10^{7}	80.0 s
Nelsen12[†]	$(2, 5)$	1.1×10^{-14}	5	7.6×10^{7}	4.5 s
Nelsen13[†]	$(2, 5)$	2.8×10^{-15}	15	1.1×10^{8}	181.2 s
Nelsen17[†]	$(1, 3)$	2.9×10^{-15}	15	7.3×10^{7}	158.6 s

The scaling column reveals the core problem with symbolic differentiation: the expression size depends on the generator's algebraic complexity in ways that do not track with dimension alone. Clayton's power-law generator and AMH's rational generator remain tractable at K=30 with 5.1×10^5 and 2.4×10^7 leaves respectively, but every other family blows up exponentially—Joe reaches 108 million leaves at K=15, InverseGaussian 164 million, and Nelsen12 76 million at K=5. Nelsen12 times out at K=10; Nelsen13 and Nelsen17 reach K=15 before timing out at the same exponential rate. Our polynomial-powering algorithm avoids this family-dependent cost: it processes all ten families in $O(d^3)$ operations regardless of the generator's algebraic form, because the cost depends only on the polynomial degree K, not on the symbolic complexity.

C Log-Space Taylor-Mode Differentiation

The log-domain machinery of Appendix B keeps the Bell pipeline's polynomial coefficients finite at high d, but it operates on *outputs* of `jet_array.jet` rather than inside the jet expansion itself. For Frank and Joe generators on extreme-tail leaves, that boundary is exactly where the failure now appears: the forward jet returns finite-but-denormal Taylor coefficients, and reverse-mode AD then materialises $1/c$ for those denormal c, overflowing to ∞ and producing $\infty \cdot 0$ in the next DOT_GENERAL step. We resolve the failure by representing every Taylor coefficient as a $(\text{sign}, \log|c|)$ pair end-to-end; the *log-space jet*, opt-in via `compile_model(..., log_jet=True)`.

C.1 Why naive float64 fails for Frank-type generators

For an Archimedean generator that decays as $\psi(t) \approx c\,e^{-t}$ at large t (Frank, Joe), the k-th Taylor coefficient at the composed argument t_r behaves like $c\,e^{-t_r}/k!$ times a polynomial in k. Once $t_r \gtrsim 500$ — which for d=75 flat copies of u=10^{-3} gives $t_r = -\sum \log u_\ell \approx 518$ — the high-order coefficients drop into float64 denormal range ($|c_k| < 2.2 \times 10^{-308}$) by $k \gtrsim 60$.

The forward computation tolerates the denormals. Backward does not: the upstream rules in `jet_array` (notably `_log_taylor` and the body of `_div_taylor_rule`) recursively divide by c in their Faà di Bruno recurrences, and the Jacobian-vector product of $1/c$ at $c \sim 10^{-310}$ is $\sim 10^{310}$, beyond `float64.max`. The resulting ∞ then meets a structural zero — for example, the constant divisor's higher Taylor terms in u/θ — and DOT_GENERAL returns NaN. The minimal reproducer is d=75 flat-Frank with all leaves at u=10^{-3}; raw forward log-density is 39.92, raw gradient is NaN.

C.2 Architecture: (sign, $\log|c|$) throughout

We replace the raw float series in `jet_array.jet` with a `LogSeries` pair (σ, λ) whose entries satisfy $c = \sigma \cdot e^{\lambda}$ for $\sigma \in \{-1, 0, +1\}$ and $\lambda \in \mathbb{R}$. Structural zero encodes as $(\sigma, \lambda) = (0, -\infty)$, which acts as the additive identity $\mathbf{0}$. Multiplication and division become sign products and log-magnitude addition or subtraction; addition uses signed logsumexp on the non-zero support, with $\mathbf{0} \oplus x = x$ by definition:

$$(\sigma_a, \lambda_a) \oplus (\sigma_b, \lambda_b) \;=\; \big(\text{sign}(S),\ \lambda_M + \log|S|\big), \quad \lambda_M = \max_{i \in A^*} \lambda_i, \quad S = \sum_{i \in A^*} \sigma_i\, e^{\lambda_i - \lambda_M},$$

where $A^* = \{i \in \{a, b\} : \sigma_i \neq 0\}$ is the active support; for empty A^* the result is $\mathbf{0}$, and a singleton reduces to its sole operand by inspection. Inside each jet rule's scan body — division, log, log1p, exp, expm1, integer power, and multiplication — the same recurrence runs, but every arithmetic operation dispatches to its log-space sibling. Coefficients with magnitude $10^{\pm 300}$ now occupy normal float64 territory in λ ($\lambda \approx \pm 690$), so no JVP rule ever forms $1/c$ for a denormal c.

C.3 Crossing the bell boundary without leaving log-space

A naive *raw on entry, log-space inside, raw on exit* wrapper re-creates the failure: `log_to_raw` materialises $\sigma \cdot e^{\lambda}$ at the boundary, and any genuinely-tiny λ becomes a denormal float that the next downstream `jnp.log(|c|)` call differentiates as $1/c \to \infty$. We extend `jet_array.jet` with `return_log_series=True` so the output stays as a `LogSeries` pair, and rewrite `bell._root_assembly` to consume that pair directly: the existing signed-logsumexp reduction reads λ as $\log|c|$ and σ as the sign, skipping the `jnp.log(jnp.abs(·))` step that was the residual NaN site. The bell pipeline's earlier stages already operate in log domain via `_normalize_poly` and the carried `total_log_scale`, so no further bell changes are needed.

C.4 Validation

- *Equivalence on benign inputs.* At $d \in \{10, 20, 50\}$ with moderate u, `log_jet=True` matches the raw path on forward log-density and gradient to relative error 10^{-10}.
- *Stability where raw fails.* At d=75 flat-Frank with all u=10^{-3}: raw gradient NaN, log-jet gradient 29.70 (finite). At the nested 11-by-9 sp500-style pattern with five extreme leaves per sector: raw gradient NaN, log-jet gradient finite; forward log-density agrees to relative error 10^{-7}.

- *Production fit.* S&P 500 d=98 Frank — the original failure that motivated this appendix — converges in 15 L-BFGS-B function evaluations to NLL $= -25749.51$ with `log_jet=True`, improving on Clayton (-25346.78) at the same dimension while remaining behind Gumbel (-26532.01). Before the fix, the run recorded `converged=False` with 44 evaluations and NLL $= -25897.14$, a fictitious value the bell pipeline's old NaN-mask had silently swallowed.
- *Regression.* The `acopula` test suite passes with `log_jet=False` producing identical bytes; the additional `tests/test_log_jet.py` (six tests) and `tests/test_log_space.py` (twenty-two tests on the `LogSeries` primitives) pass under JIT.

C.5 Cost and configurability

The log-space rules add one scalar field to every Taylor coefficient and one sign-tracking branch in each scan body, so memory doubles and per-iteration arithmetic roughly doubles versus the raw path. Compile time grows by 30–50% on the d=98 Frank fit (30 seconds cold versus 18 seconds for the raw d=75 case). Steady-state per-evaluation runtime grows by a similar factor. The kwarg defaults to `False`, so existing callers pay nothing; users of high-d Frank or Joe pass `log_jet=True` once and the rest of the pipeline adapts.

D Polynomial Multiplication Alternatives

We chose direct $O(K^2)$ convolution over the textbook $O(K \log K)$ FFT path because FFT loses on both speed and precision in the regime copula applications occupy. Our implementation multiplies truncated polynomials with `lax.conv_general_dilated`, which executes a direct convolution. Table 10 confirms the choice empirically: direct convolution beats rfft-based multiplication for every sector size up to K=192 and only loses at K=256, far above the regime where copula applications operate.

Table 10: Wall-clock time per truncated polynomial-powering scan ($p^1, \ldots, p^K$) at sector size K, JIT-compiled in JAX on a single CPU thread. Direct convolution wins by a factor of 2–5 for every $K \leq 192$; the FFT path's relative error is unbounded past K=64, with the earlier K=16 spike ($\sim$0.02) and K=24 spike ($\sim$0.56) reflecting the alternating-sign cancellation that destabilises the convolution before the magnitude blow-up.

K	direct (ms)	FFT (ms)	direct speedup
4	0.007	0.014	2.01×
16	0.020	0.067	3.29×
32	0.030	0.152	5.08×
64	0.060	0.331	5.47×
128	0.366	0.796	2.18×
192	1.349	1.783	1.32×
256	5.443	2.748	0.50× (FFT wins)

Three observations support this choice. First, polynomial degrees in copula applications rarely exceed $K = 50$, where FFT's constant overhead—complex-arithmetic promotion, bit-reversal permutations—wipes out its asymptotic savings. Second, the short-kernel optimisation in `_convolve_short_kernel` already reduces the Cauchy-product step from $O(d_v^2)$ to $O(d_v \cdot d_{\max}(v))$, narrowing the gap further. Third—decisively for our regime—FFT-based polynomial multiplication satisfies only a *norm-wise* forward-error bound [Higham, 2002, Ch. 24], $|\mathrm{fl}(a * b)_k - (a * b)_k| \leq c\,\epsilon \log K \, \|a\|_2 \|b\|_2$, so every output entry inherits an absolute error proportional to the *largest* coefficient in the inputs. Direct convolution, by contrast, preserves *component-wise* precision: each output entry's error scales with the coefficients that actually sum into it. Bell polynomial coefficients within a single β-vector span many orders of magnitude—factorial growth from β_1 to β_{d_v} gives dynamic range above 10^{60} already at $d_v = 50$, far exceeding the $\sim 10^{16}$ resolution of float64—so the small leading coefficients that the nested-density formula multiplies against $\psi_r^{(k)}$ at low k lose all digits to FFT round-off. The repeated power scan compounds this loss at every step, so FFT-based powering retains no usable precision well below K=100 in float64 unless the implementation first rescales the polynomial into a narrow dynamic range—a rescaling that our log-domain pipeline already performs

for the direct path at zero asymptotic cost. Should an application drive K above 200, swapping in an FFT path requires changing only two internal helpers, with no public-API impact.

Alternative Bell-polynomial algorithms. Factorisation-based algorithms compute partial Bell polynomials via recurrence relations or matrix factorisations, achieving the same $O(d^3)$ asymptotic complexity with different constants. Our polynomial-powering formulation sidesteps these alternatives because Taylor-mode AD already produces jet coefficients in exactly the truncated-power-series representation that polynomial powering consumes, so no format conversion is needed. Graph-level AD optimisations such as cross-country elimination [Bettencourt et al., 2019] can reduce compilation overhead but remain orthogonal to the density-level contribution of this work.

E Symbolic Inversion and Numeric Fallback

Two paths supply the inverse generator ψ^{-1} that the density and sampler require: a symbolic derivation via Oryx for closed-form generators, and a numeric bisection fallback for everything else. Oryx [Radul et al., 2023] derives ψ^{-1} by tracing the generator through JAX's intermediate representation and pattern-matching each primitive against a table of known algebraic inverses—for example, $\exp \leftrightarrow \log$, $x \mapsto x^a \leftrightarrow x \mapsto x^{1/a}$, and $x \mapsto x + c \leftrightarrow x \mapsto x - c$. The procedure succeeds whenever the generator is a composition of individually invertible operations with no data-dependent control flow. All ten built-in families satisfy this condition.

Oryx's pattern matcher cannot invert functions that contain data-dependent branching—`lax.cond` or `lax.switch`—because the branch the tracer follows at trace time may differ from the branch the runtime selects at evaluation time. Functions that call numerical subroutines such as adaptive quadrature or iterative solvers also fail, because those routines expand into opaque `while_loop` primitives. The matcher raises an error at trace time in either case, never silently returning an incorrect result.

The framework probes each generator at initialisation by evaluating the Oryx-derived inverse on a concrete test value. When the probe succeeds, the symbolic inverse handles all subsequent calls. When it fails, the framework switches transparently to a numeric path that solves $\psi(s) = t$ for s via `optimistix.Bisection` with tolerance 10^{-12}. Because generators are strictly decreasing on $[0, \infty)$, the bisection bracket always exists. Gradients propagate through the root-finding step via the implicit function theorem: `optimistix.ImplicitAdjoint` differentiates through the solution by solving $\frac{\partial \psi}{\partial s}\big|_{s^\star} \delta s = \delta t$. Our test suite verifies round-trip accuracy—$|\psi(\psi^{-1}(t)) - t| < 10^{-10}$—and gradient correctness against the analytic derivative.

F Rosenblatt Sampling via Generator Derivatives

The same Bell-polynomial machinery that delivers the density also samples from a nested Archimedean copula for free. The Bell-polynomial recursion of Appendix A supplies, as a byproduct, the high-order generator-composition derivatives that the textbook *conditional-distribution method*—also called Rosenblatt sampling—requires. The same sampler draws the synthetic data that validates the Frank and Gumbel parameter-recovery results in Appendix M.1, and samples the InverseGaussian†, Nelsen9†, Nelsen12†, Nelsen13†, and Nelsen17† families used in the retinopathy experiment (Section 5.6) with no per-family implementation.

The conditional-distribution method. Sampling from an Archimedean copula $C(u_1, \ldots, u_d) = \psi\big(\psi^{-1}(u_1) + \cdots + \psi^{-1}(u_d)\big)$ proceeds variable by variable. Let $t_j = \sum_{k=1}^{j} \psi^{-1}(u_k)$ denote the running sum of inverse-generator transforms. The conditional distribution of u_j given $u_1, \ldots, u_{j-1}$ has the closed form

$$C_{j|1..j-1}(u_j) \;=\; \frac{\psi^{(j-1)}(t_j)}{\psi^{(j-1)}(t_{j-1})}, \tag{15}$$

which depends only on the $(j-1)$-th derivative of the generator evaluated at two arguments. The sampler draws $V_j \sim \mathrm{Uniform}(0, 1)$ independently for each variable, then inverts Equation (15) for u_j satisfying $C_{j|1..j-1}(u_j) = V_j$ by bisection on u_j. The nested case extends the same recursion through the tree by evaluating the appropriate edge-composition derivative at each visited internal

node. Algorithm 4 gives the flat-copula sampler. The nested case generalises by replacing the single-generator derivative $\psi^{(j-1)}(\cdot)$ with the full Bell-polynomial mixed partial of Algorithm 1: each conditional CDF $C_{j|1..j-1}(u_j)$ is a ratio of two partial CDFs that differentiate variables $1,\dots,j-1$ and fix variables $j,\dots,d$ to specific values, with $u_k = 1$ marginalising the unsampled tail. The censoring-mask machinery that already drives likelihood evaluation supplies exactly this primitive, so the nested sampler reuses the density code path twice per variable (Algorithm 5).

Algorithm 4 Rosenblatt sampler for an Archimedean copula with generator ψ in dimension d.

Require: Generator ψ, dimension d, RNG
Ensure: Sample $u \in (0,1)^d$
Draw $V_1,\dots,V_d \sim \mathrm{Uniform}(0,1)$ iid
$u_1 \leftarrow V_1$; $t_1 \leftarrow \psi^{-1}(u_1)$
for $j = 2,\dots,d$ **do**
  Define $G_j(u) := \psi^{(j-1)}\big(t_{j-1} + \psi^{-1}(u)\big) \,/\, \psi^{(j-1)}(t_{j-1})$ ▷ $\psi^{(j-1)}$ *via Taylor jet*
  $u_j \leftarrow \textsc{Bisect}(G_j(u) = V_j,\ u \in (0,1))$
  $t_j \leftarrow t_{j-1} + \psi^{-1}(u_j)$
return $(u_1,\dots,u_d)$

Algorithm 5 Rosenblatt sampler for a nested Archimedean copula on tree T. Each inner call invokes the Bell-polynomial mixed partial of Algorithm 1 with a mask that differentiates only the already-sampled variables. An efficient implementation memoizes, at every internal node v, the Bell-polynomial state of Algorithm 1: the running sum t_v, the count d_v of leaves already sampled in v's subtree, and the list of completed children's $\alpha^{(c)}$ vectors; each per-leaf step then traces only from the leaf's parent up to the root.

Require: Nested copula tree T, leaf order $u_1,\dots,u_d$, RNG
Ensure: Sample $u \in (0,1)^d$
Draw $V_1,\dots,V_d \sim \mathrm{Uniform}(0,1)$ iid
for $j = 1,\dots,d$ **do**
  $\delta_j \leftarrow (\underbrace{1,\dots,1}_{j-1},\ 0,\dots,0)$ ▷ *differentiate* $u_1,\dots,u_{j-1}$ *only*
  Define $D_j(u) \leftarrow \textsc{BellPartial}(T,\ (u_1,\dots,u_{j-1},\ u,\ 1,\dots,1),\ \delta_j)$ ▷ *partial CDF*
  $u_j \leftarrow \textsc{Bisect}(D_j(u) \,/\, D_j(1) = V_j,\ u \in (0,1))$
return $(u_1,\dots,u_d)$

Connection to prior work. Hofert [2008] identified the high-order generator-composition derivatives in Equation (15) as the principal obstacle to applying this method beyond small dimensions and developed Marshall–Olkin frailty sampling as the practical alternative. That alternative requires a known Laplace transform per family, which blocks cross-family nesting and rules out any generator without a closed-form mixing distribution—for example, the Gen-AC generator of Ng et al. [2021] adopted by Liu et al. [2025b]. The Bell-polynomial recursion of Appendix A computes exactly these derivatives in closed form for any user-supplied generator with the same complexity profile as the density evaluation, so sampling and likelihood share a single code path.

G Validity of Nested Constructions

A nested Archimedean copula defines a valid distribution only when every parent–child generator pair satisfies a sufficient nesting condition. McNeil [2008] required the composition $h = \psi_{\text{outer}}^{-1} \circ \psi_{\text{inner}}$ to be a *Bernstein function* (i.e., its derivative completely monotone): $(-1)^{k-1} h^{(k)}(t) \geq 0$ for all $k \geq 1$ and $t \geq 0$. McNeil and Nešlehová [2009] subsequently characterised d-dimensional Archimedean copulas through the Williamson d-transform, weakening the complete-monotonicity requirement on the generator itself to d-monotonicity and setting the foundation for the results below. Rezapour [2015] showed that a weaker condition suffices: h need only be d_c-*monotone*, that is, $(-1)^{m-1} h^{(m)}(t) \geq 0$ for $m = 1,\dots,d_c$ and all $t \geq 0$, where d_c counts uncensored leaves under the child node. This finite condition matches the information our algorithm already computes: the JET call at each edge returns exactly d_c Taylor coefficients of h.

For the classical families, closed-form sufficient conditions exist: Clayton and Gumbel both require $\theta_{\text{outer}} \leq \theta_{\text{inner}}$, Frank requires the same ordering, and Joe requires $1 \leq \theta_{\text{outer}} \leq \theta_{\text{inner}}$. These orderings guarantee complete monotonicity and therefore d_c-monotonicity.

Our framework does *not* enforce these constraints automatically. For same-family pairs, a delta parameterisation $\theta_{\text{inner}} = \theta_{\text{outer}} + \text{softplus}(\delta)$ enforces the ordering by construction; all same-family experiments use this approach (Table 11).

Table 11: Nesting conditions for generator pairs in our experiments.

Experiment	Outer	Inner	Condition	Enforcement
S&P / MIMIC-IV	Clayton	Clayton	$\theta_{\text{outer}} \leq \theta_{\text{inner}}$	delta param.
S&P / MIMIC-IV	Frank	Frank	$\theta_{\text{outer}} \leq \theta_{\text{inner}}$	delta param.
S&P / MIMIC-IV	Gumbel	Gumbel	$\theta_{\text{outer}} \leq \theta_{\text{inner}}$	delta param.
MIMIC-IV	AMH	Clayton	$\theta_{\text{inner}} \geq 1$ [Hofert, 2010]	shifted softplus
Retinopathy	all 10 families		single-layer: nesting N/A	

Same-family rows enforce $\theta_{\text{outer}} \leq \theta_{\text{inner}}$ via the delta parameterisation. The AMH/Clayton row uses Hofert [2010]'s closed-form constraint $\theta_{\text{Clayton}} \geq 1$, enforced by shifted-softplus $\theta_{\text{Clayton}} = 1 + \text{softplus}(\delta)$; this is the single documented closed-form cross-family Archimedean pair with a McNeil-2008 Bernstein-function nesting proof. InverseGaussian, Nelsen13, and Nelsen17 appear only in the retinopathy experiment, which uses single-layer bivariate copulas; nesting conditions do not apply.

Cross-family and custom generators. The closed-form constraints above cover the families used in our experiments, but a user nesting two different families—or supplying a custom generator—needs a more general route. When no closed-form parameter constraints exist, enforcing the nesting condition during maximum likelihood estimation becomes a *semi-infinite program* (SIP). The d_e-monotone condition must hold at every $t \geq 0$, producing infinitely many inequality constraints over the finite-dimensional parameter vector $\boldsymbol{\theta} \in \mathbb{R}^p$ stacking all per-edge generator parameters. Let $\mathcal{E}$ denote the set of parent–child edges in the copula tree, d_e the number of uncensored leaves under the child of edge e, and $h_e(t; \boldsymbol{\theta}) = \psi_{\text{outer}}^{-1}\big(\psi_{\text{inner}}(t)\big)$ the edge composition, with $h_e \equiv h_{vc}$ when $e = (v, c)$. The constrained MLE is

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & \ell(\boldsymbol{\theta}; \mathbf{X}) \\ \text{s.t.} \quad & (-1)^{m-1}\, h_e^{(m)}(t; \boldsymbol{\theta}) \;\geq\; 0 \quad \forall\, e \in \mathcal{E},\;\; m = 1, \ldots, d_e,\;\; t \geq 0. \end{aligned} \tag{16}$$

Grid discretisation. A standard approach to a SIP replaces the continuum of constraints with a finite grid $\{t_1, \ldots, t_G\} \subset [0, t_{\max}]$ of G user-chosen points:

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & \ell(\boldsymbol{\theta}; \mathbf{X}) \\ \text{s.t.} \quad & (-1)^{m-1}\, h_e^{(m)}(t_g; \boldsymbol{\theta}) \;\geq\; 0 \quad \forall\, e \in \mathcal{E},\;\; m = 1, \ldots, d_e,\;\; g = 1, \ldots, G. \end{aligned} \tag{17}$$

Checking each grid point at one edge requires a single JET call that returns the d_e Taylor coefficients of h_e; the signs of those coefficients encode the d_e-monotone condition. Concretely, the JET call returns the Taylor coefficient vector $p_m = h_e^{(m+1)}(t)/(m+1)!$ for $m = 0, \ldots, d_e - 1$. Because factorial scaling preserves sign, the d_e-monotone condition $(-1)^{m-1} h_e^{(m)}(t) \geq 0$ for $m = 1, \ldots, d_e$ reduces to checking that $(-1)^m p_m \geq 0$ for each m—an alternating-sign test on the Taylor coefficients. Each density evaluation already performs one such call per edge at $t = t_e$, so N training observations contribute N grid points per edge at no extra cost; a denser grid requires $G - N$ additional JET calls.

Computing the penalty in principle. The ACOPULA framework can compute the nesting penalty as a byproduct of the density evaluation. At each edge e, the JET call that produces the Taylor coefficients $p_0, p_1, \ldots, p_{d_e}$ for the Bell transform also provides the sign information for the diagnostic. The penalty takes the form

$$P(\boldsymbol{\theta}; \mathbf{x}) = \sum_{e \in \mathcal{E}} \sum_{m=0}^{d_e - 1} \big[\min\big(0,\; (-1)^m\, p_m\big)\big]^2,$$

evaluated at $t = t_e(\mathbf{x})$, the sector sum for the current observation $\mathbf{x}$. When $P = 0$, all Taylor coefficients satisfy the alternating-sign condition at every evaluation point encountered during training. A penalised optimiser would minimise $-\ell(\boldsymbol{\theta}) + \rho\, P(\boldsymbol{\theta})$ with penalty weight $\rho \geq 0$, but all experiments in this paper avoid this route: every nested fit uses one of the three valid-by-construction parameterisations summarised in Table 11—same-family delta, AMH/Clayton shifted-softplus, or the Gen-AC/ACNet Bernstein composition (Appendix G.1)—which makes the penalty identically zero by construction. We describe the penalty approach here as a conceptual extension for users who might want to experiment with cross-family or custom generators that lack closed-form constraints; we leave empirical validation of the penalised route to future work.

Limitations. The grid relaxation Equation (17) is necessary but not sufficient for the full SIP Equation (16): violations can hide between grid points. Certifying the d_e-monotone condition on all of $[0, \infty)$ for arbitrary generators remains an open problem.

G.1 Validity of the nested Gen-AC/ACNet construction

The nested construction used by the Gen-AC and ACNet rows of Section 5.4 and by the HACSurv port of Appendix T pairs a shared outer generator $\psi_{\text{out}}(s)$—a mixture of exponentials $L_{\text{out}}^{-1} \sum_{l=1}^{L_{\text{out}}} \exp(-M_l s)$ for Gen-AC (at L_{out} MC nodes) and the softmax-convex cascade of Appendix I for ACNet—with per-panel inner $\psi_{\text{in},k}(t) = \psi_{\text{out}}(b_k(t))$ and inner Bernstein function $b_k(t) = e^{\mu_k}\, t + e^{\beta_k}\big(1 - L_{\text{in}}^{-1} \sum_{l=1}^{L_{\text{in}}} e^{-N_{k,l} t}\big)$, where $L_{\text{out}}, L_{\text{in}} \in \mathbb{N}$ are the outer and inner mixture sizes, $M_l, N_{k,l} \geq 0$ are exponential rates, k indexes the per-panel inner generator, and $\mu_k, \beta_k \in \mathbb{R}$ are scalar log-shift parameters. The construction is valid for any $(N_k, \mu_k, \beta_k) \in \mathbb{R}_{\geq 0}^{L_{\text{in}}} \times \mathbb{R}^2$, and the optimiser need not enforce any constraint. The argument runs in three steps. First, b_k is a Bernstein function: $b_k(0) = 0$, $b_k > 0$, and b_k' is completely monotone (CM) as a sum of CM components. Second, the composition of a CM function with a Bernstein function is itself CM [McNeil, 2008, Rezapour, 2015], so $\psi_{\text{in},k}$ is a valid generator. Third, $\psi_{\text{out}}^{-1} \circ \psi_{\text{in},k} = b_k$ is automatically Bernstein, so the parent–child pair satisfies the nesting condition.

H Gen-AC MLP-pushforward generator with Fourier features

The Gen-AC rows of Table 4 implement the MLP-pushforward generator of Ng et al. [2021] with frailty $V = \exp(\text{MLP}(U))$, $U \sim \text{Uniform}(0, 1)$, so the Laplace transform is $\psi(t) = \mathbb{E}_U[\exp(-Vt)]$. HACSurv [Liu et al., 2025b] uses this construction directly with a three-layer MLP of hidden width 10 and leaky-ReLU activations; the row labelled Gen-AC$_{1.1\text{k}}$ reproduces that architecture with one outer MLP and a per-panel inner MLP, composed with the Bernstein drift of Appendix G.1, for $7 \cdot 2 + 8 \cdot 141 = 1{,}142$ parameters. We evaluate $\mathbb{E}_U$ at $N_z{=}32$ fixed equispaced nodes on $(0, 1)$; equispaced nodes instead of per-step Monte-Carlo resamples remove the Jensen bias and shift held-out NLL by under 0.04 nats.

The raw-u input of HACSurv's MLP plateaus at held-out NLL 7.248—worse than Clayton's 7.088 by a margin that separates under every correction we apply (Appendix S.1.2)—because a small MLP of Lipschitz constant ≈ 1 cannot place neighbouring $\log M_i$ at widely separated rates: the 32 atoms collapse onto a narrow band. We lift that prior by replacing the scalar input with Fourier features [Tancik et al., 2020] at K frequencies:

$$u \;\longmapsto\; \big[\cos(2\pi k u),\, \sin(2\pi k u)\big]_{k=1}^{K} \;\in\; \mathbb{R}^{2K},$$

passed into an otherwise identical three-layer MLP of hidden width h. The Fourier basis lets the network output sharply different $\log M_i$ at neighbouring u_i without requiring a large Lipschitz constant, recovering the flexibility that direct trainable $\log M \in \mathbb{R}^{32}$ would have. Table 4 reports two configurations: a parameter-matched variant Gen-AC-Fourier$_{260}$ at $(K{=}2, h{=}3)$, and the full model Gen-AC-Fourier$_{3.6\text{k}}$ at $(K{=}16, h{=}10)$. At matched capacity Gen-AC remains less efficient than ACNet in this cohort—at 260 parameters Gen-AC-Fourier$_{260}$ reaches 7.165 versus ACNet's 7.091 at 70 parameters, and of that pair only Gen-AC-Fourier$_{260}$ separates from the Clayton/Clayton reference, at all five splits and across them. Releasing the parameter budget closes the full Fourier model to 0.011 nats of the best ACNet, near enough that the all-pairs band at the published split leaves it, all three ACNet rows, and Clayton/Clayton indistinguishable from the best-scoring model on that split (Appendix S.1.2).

I Deep ACNet Generator

The ACNet rows of Table 4 use the depth-L softmax-convex-combination generator of Ling et al. [2020] as the outer ψ_{out} and the Bernstein-composition inner $\psi_{\text{in},k}(t) = \psi_{\text{out}}(b_k(t))$ from Appendix G.1. Given widths $(w_0, \ldots, w_{L-1})$ and a query $t \in [0, \infty)$, the generator is the terminal state of an L-layer cascade of positive exponential shifts interleaved with simplex-constrained convex combinations, defined recursively as

$$\begin{aligned} F_i^{(0)} &= e^{-s_{0,i}t}, \\ F_i^{(\ell)} &= e^{-s_{\ell,i}t} \cdot \left(A_{\ell-1} F^{(\ell-1)}\right)_i \text{ for } \ell = 1, \ldots, L-1, \\ \psi_{\text{out}}^{\text{ACNet}}(t) &= A_{L-1} F^{(L-1)}, \end{aligned}$$

where $s_\ell = e^{\tilde{s}_\ell} \in \mathbb{R}^{w_\ell}$ are per-layer positive exponential rates and $\tilde{s}_\ell$ their unconstrained log parameters, $A_\ell \in \mathbb{R}^{w_{\ell+1} \times w_\ell}$ for $\ell < L-1$ and $A_{L-1} \in \mathbb{R}^{1 \times w_{L-1}}$ are the row-wise softmaxes of free matrices $\tilde{A}_\ell$ so that every row of A_ℓ lies on the simplex. The first layer's convex combination acts on the scalar 1, so the recursion absorbs it into the initialisation $F^{(0)}$. Each per-layer factor $e^{-s_{\ell,i}t}$ is a completely monotone function of t, every row of A_ℓ is a convex combination, and closure of complete monotonicity under convex combinations and under products of completely monotone functions (applied at every step of the recursion) yields a scalar-valued completely monotone function, so $\psi_{\text{out}}^{\text{ACNet}}$ is a valid Archimedean generator. The parameterisation imposes no constraint, and the optimiser updates $\tilde{s}_\ell, \tilde{A}_\ell$ freely.

The inner generator follows the same Bernstein-composition recipe as Appendix G.1, $\psi_{\text{in},k}(t) = \psi_{\text{out}}^{\text{ACNet}}(b_k(t))$ with $b_k(t) = e^{\mu_k}\, t + e^{\beta_k}(1 - L_{\text{in}}^{-1} \sum_l e^{-N_{k,l}t})$; the same Schoenberg-theorem argument shows $\psi_{\text{in},k}$ is completely monotone and the parent–child composition $\psi_{\text{out}}^{-1} \circ \psi_{\text{in},k} = b_k$ is Bernstein. The compact-parameter count in Table 4's $|\theta|$ column sums the per-layer unconstrained $\tilde{s}_\ell \in \mathbb{R}^{w_\ell}$ and $\tilde{A}_\ell \in \mathbb{R}^{w_{\ell+1} \times w_\ell}$ of the shared outer, the scalar log-shift pair (μ_k, β_k) per panel, and the per-panel inner rates $N_k \in \mathbb{R}^{L_{\text{in}}}$.

J Detailed R Comparison

Our Bell polynomial log-density agrees with R's `nacLL` [Hofert and Pham, 2013] to machine precision on every configuration where R completes, establishing numerical correctness of the framework's forward evaluation. We compare the two implementations on 12 nested Clayton–Clayton configurations spanning $d = 4$ to 60 total dimensions with identical random input matrices, $\theta_{\text{outer}} = 2$, $\theta_{\text{inner}} = 5$, and $n = 5$ observations per configuration. Table 12 reports the results.

Table 12: Density correctness of ACOPULA versus R's `nacLL` on nested Clayton–Clayton copulas. Here, s denotes the number of sectors, K the leaves per sector, and $d = sK$ the total dimension. The relative error is $|\ell_{\text{ours}} - \ell_R|/|\ell_R|$. Arbitrary-precision verification attributes the d=60 discrepancy to R's `nacLL`, not our framework.

s	K	d	Rel. error	Status
2	2	4	$< 10^{-14}$	✓
2	5	10	$< 10^{-14}$	✓
2	10	20	$< 10^{-14}$	✓
2	20	40	$< 10^{-14}$	✓
2	30	60	3.5×10^{-5}	R precision loss
3	5	15	$< 10^{-14}$	✓
4	5	20	$< 10^{-14}$	✓
5	5	25	$< 10^{-14}$	✓
5	10	50	$< 10^{-14}$	✓
6	5	30	$< 10^{-14}$	✓
7	5	35	$< 10^{-14}$	✓
8	5	40	$< 10^{-14}$	✓

Eleven of 12 configurations agree to relative error below 10^{-14}, matching float64 roundoff. We attribute the single d=60 discrepancy to precision loss in R's `nacLL`, not our framework: Mathematica's

symbolic differentiation of the copula CDF via `D[C, u1, ..., ud]` provides an algorithm-agnostic ground truth that agrees with our framework to machine epsilon at every K from 2 to 30, while R's `nacLL` diverges at $K \geq 31$ (see also Appendix B for the cross-family Mathematica check). The scripts `mathematica_ground_truth.wls` and `verify_against_mathematica.py` reproduce this comparison.

Controlled runtime comparison. Under matched single-thread settings, ACOPULA runs orders of magnitude faster than R's `nacLL` at moderate dimension and completes families that `nacLL` cannot evaluate at all. Table 13 reports wall-clock time for the log-likelihood of n=100 observations under an apples-to-apples single-thread configuration that pins thread counts and the BLAS backend across both runtimes. Both R and Python run with `OMP_NUM_THREADS=OPENBLAS_NUM_THREADS=MKL_NUM_THREADS=1`, read the same float64 input matrices from disk, and time identical nested same-family configurations with θ_{outer}=2 and θ_{inner}=5 via `microbenchmark` on the R side and `time.perf_counter` after a JIT warmup on the Python side. We compare three families that R's `nacLL` purports to support: Clayton, Frank, and Gumbel.

Table 13: Controlled single-thread runtime of ACOPULA, JIT-compiled on CPU, versus R `nacLL` on nested same-family copulas with n=100 observations, θ_{outer}=2, and θ_{inner}=5. The "Compile" column reports the one-time JIT cost; N/A indicates that R's `nacLL` aborts on the configuration.

Family	d	ACOPULA (ms)	Compile (ms)	R `nacLL` (ms)	Speedup
Clayton	4	0.73	3,240	1.95	2.7×
Clayton	10	1.69	3,766	6.04	3.6×
Clayton	15	1.58	3,766	19.61	12.4×
Clayton	20	2.19	4,768	81.70	37.3×
Clayton	25	2.39	5,103	448.65	187.4×
Clayton	30	2.57	5,854	3,137.66	1218.7×
Frank	4	1.25	3,339	N/A	—
Frank	10	1.99	3,676	N/A	—
Frank	15	2.01	4,364	N/A	—
Frank	20	2.76	5,219	N/A	—
Frank	25	3.18	5,622	N/A	—
Frank	30	3.47	6,199	N/A	—
Gumbel	4	0.94	2,033	3.71	3.9×
Gumbel	10	1.80	2,370	11.86	6.6×
Gumbel	15	1.55	2,954	29.12	18.8×
Gumbel	20	2.25	3,480	96.97	43.1×
Gumbel	25	2.75	3,916	492.29	179.1×
Gumbel	30	4.21	4,292	3,106.20	737.4×

Three findings emerge from Table 13. First, our framework matches R's log-likelihood values to relative error below 1.7×10^{-15} in every cell where R completes, ruling out a precision artefact in the timing gap. Second, the speedup grows from $\sim$4× at d=4 to over 700× at d=30 for Gumbel: R's `nacLL` scales exponentially in the number of sectors via partition enumeration, whereas our polynomial powering scales as $O(d^3)$, so the gap widens monotonically with d. Third, R's `nacLL` aborts on every Frank configuration with the error `a.coeff: Frank's family is currently not supported`; its documented support for nesting across the five classical families therefore fails for Frank, AMH, and Joe, because `nacLL` hard-codes the partial-Bell coefficients only for Clayton and Gumbel. Table 14 aggregates the structural differences that drive these gaps.

Two practical caveats sharpen the comparison. R's closed-form evaluations are faster per call than our jet-based k-step scan, but the Stirling and Eulerian numbers they rely on overflow 64-bit floats at moderate orders. Our JIT-compiled pipeline pays a one-time compilation cost of $\sim$50 ms and thereafter evaluates in < 0.25 ms per observation. The `Copula` base class also exposes an optional `log_generator_kth_derivative(t, k)` hook for users who have a closed-form $\psi^{(k)}$, recovering R's per-call speed while retaining full framework compatibility.

Polynomial scaling in s. Our Cauchy-product algorithm replaces R's exponential partition enumeration with polynomial multiplication, so cost in the number of sectors s grows polynomially rather

Table 14: Comparison of ACOPULA with R's `copula`/`nacopula` packages.

	R copula/nacopula	**ACOPULA (ours)**
Derivatives	Hand-derived closed forms; must implement $\psi^{(k)}$	Automatic via jet; only ψ needed
New family	ψ + all derivatives in C/R	ψ in 3 lines of Python
Param. gradients	Analytical for built-in families; numerical otherwise	Exact via `jax.grad` (end-to-end)
Compilation	Interpreted R + C extensions	XLA JIT compilation
Censoring	Custom per-family code	Automatic: `censored=True`
Bell polynomials	Per-family Stirling-number tables	Polynomial powering of jet output
Nested density	2-level only; Clayton/Gumbel only	Arbitrary depth and all families

than exponentially. To quantify the difference, we time the JIT-compiled log-likelihood for nested Clayton copulas while varying s from 2 to 50 at sector size $K = 2$, for which $d = 2s$. Execution time grows from 0.4 ms at s=2 to 72 ms at s=50, matching approximately $O(s^{1.7})$ scaling. Repeating with $K = 5$ up to s=30, with $d = 5s$, yields a similar polynomial exponent. By contrast, R's partition count exceeds 10^{15} at $s = 50$, rendering exact enumeration infeasible. Beyond raw scaling, the choice between exact and Monte Carlo $\psi^{(k)}$ evaluation involves further design tradeoffs that we discuss next.

Extended Gen-AC comparison. Our Taylor-mode computation of $\psi^{(k)}$ and the Monte Carlo estimation in Gen-AC [Ng et al., 2021] occupy complementary points in the exactness–scalability tradeoff. Taylor-mode AD computes each $\psi^{(k)}$ exactly in $O(k)$ time per coefficient, yielding machine-precision likelihoods that support gradient-based parameter estimation, nested copula structures, and per-variable censoring without approximation error. Gen-AC, in contrast, sidesteps generator differentiation by representing $\psi^{(k)}$ as a Monte Carlo average over learned latent variables; this representation scales to very high k with bounded per-sample cost but introduces sampling variance and restricts the generator to the learned Laplace class. For the moderate dimensions typical of actuarial and survival-analysis applications, $d \leq 100$, the exact approach dominates: it produces unbiased gradients and handles arbitrary user-defined generators. Gen-AC suits regimes where only samples—not likelihoods—matter, or where d is so large that even $O(k)$ exact derivative cost grows prohibitive.

Bivariate censored comparison with CopulaCenR. ACOPULA reproduces CopulaCenR's parameter estimates on right-censored bivariate data and succeeds where CopulaCenR's finite-difference optimiser fails. We compare against CopulaCenR [Sun and Ding, 2020], an R package that fits bivariate copulas to right-censored data via joint MLE with parametric marginals. On the retinopathy dataset, with 197 patients at 60.7% censoring, CopulaCenR's Clayton fit yields $\hat{\eta} = 0.895$ versus our two-stage Inference Functions for Margins (IFM) [Joe and Xu, 1996, Genest et al., 1995] estimate $\hat{\theta} = 0.897$, a 0.3% difference; its Gumbel fit yields $\hat{\eta} = 1.256$ versus our $\hat{\theta} = 1.249$, a 0.6% difference. The CopulaCenR estimate $\hat{\eta}$ uses the same Clayton/Gumbel parameterisation as our $\hat{\theta}$, so the two are directly comparable. CopulaCenR's Frank optimiser fails on this dataset with a finite-difference overflow, while our AD-based gradient computation converges to $\hat{\theta} = 2.255$. The close parameter agreement on two families confirms that the copula likelihoods agree in the bivariate censored setting, despite two methodological differences: joint MLE versus IFM, and shared versus per-margin Weibull marginals.

K Relationship to Copulas.jl

Copulas.jl [Laverny and Jimenez, 2024] is the closest prior tool to ACOPULA: it evaluates flat, single-layer Archimedean copula densities by the same Taylor-mode AD over the generator, and Table 1 reflects its state at commit `aeb7611c`, before it supported nesting. After the first preprint of this work, a collaboration with the package's author fixed earlier limits on per-variable censoring and high-dimensional precision, which together enable per-variable censoring at high dimension. We then ported the nested-tree algorithm of this work into the package: our pull request #367, merged

in July 2026, adds it as a new feature, and with it Copulas.jl reaches the same arbitrary nesting depth as ACOPULA. For nested classical-generator densities the two libraries then differ only in numerics—Copulas.jl evaluates in Float64 by default and in BigFloat on request, while ACOPULA evaluates in Float64 log-space—and in GPU support, which ACOPULA provides through JAX. On the d=53 MIMIC-IV tree, the censored log-densities of the two libraries agree to 8.2×10^{-13} (Appendix N). Copulas.jl in principle admits deep Archimedean generators, but no implementation exercises them at the time of writing.

L AD Gradient Verification

JAX's reverse-mode AD produces parameter gradients that match central finite differences to relative error below 10^{-8} across families, dimensions, and censoring, validating the gradient path that drives parameter estimation. We compare the AD gradient $g_{\mathrm{AD}} = \nabla_\theta \ell$ (where ℓ is the total log-likelihood across the configuration's training sample) that `jax.grad` returns against the central finite-difference approximation $g_{\mathrm{FD},i} = \big[\ell(\theta + \varepsilon e_i) - \ell(\theta - \varepsilon e_i)\big]/(2\varepsilon)$, with $\varepsilon = 10^{-5}$ and $e_i \in \mathbb{R}^p$ the i-th standard basis vector for parameter dimension p (the `Params` column in Table 15), across four nested copula configurations that exercise different families, dimensions, and censoring. Table 15 reports the maximum relative error $\max_i |g_{\mathrm{AD},i} - g_{\mathrm{FD},i}| / \max(|g_{\mathrm{AD},i}|, 10^{-10})$.

Table 15: AD gradient vs. central finite differences (ε=10^{-5}). All configurations use 2-level nesting with the stated sector structure $(M \times K)$ denoting M sectors of K leaves each.

Configuration	d	Params	Max rel. error
Clayton (2×5)	10	2	5.7×10^{-9}
Frank (2×5)	10	2	2.0×10^{-9}
Gumbel (2×5)	10	2	8.9×10^{-11}
Clayton (4×5), 55% cens.	20	2	5.9×10^{-10}

All four configurations achieve relative error below 10^{-8}, confirming that JAX's reverse-mode AD produces correct gradients through the full Bell-polynomial pipeline—including the log-domain rescaling, signed logsumexp, and polynomial powering steps that Appendix B describes. The censored configuration, with 55% of variables censored, shows no degradation relative to the fully observed cases, which validates the gradient path through the dynamic censoring mechanism.

M High-Dimensional Parameter Recovery

The MLE recovers the true parameters with bias under 2.7% and sampling variability that the observed-Fisher SE calibrates to within 2.9% on average, across $N \in \{200, 500, 1000\}$ and 500 replications per cell. Table 3 in the main text reports replicated recovery at N=500; Table 16 extends that result to the full N grid and reports bias, empirical SE, observed-Fisher SE, RMSE, and the L-BFGS-B convergence rate per cell. The empirical SE shrinks at the expected $1/\sqrt{N}$ rate as N grows: the EmpSE ratio between N=200 and N=1000 ranges from 2.03 to 2.33 across the seven parameters, all close to the theoretical $\sqrt{5} \approx 2.24$. The observed-Fisher SE matches the empirical SE within 2.9% on average and within 8.3% at worst, confirming that the Hessian-based asymptotic approximation calibrates the actual sampling variability of the MLE on the Bell-polynomial likelihood. The bias remains uniformly small, under 2.7%, and stable across N, indicating that the residual bias arises from finite-sample MLE bias rather than from convergence failures.

M.1 Frank and Gumbel Recovery

The recovery result extends to Frank and Gumbel generators, confirming that the Bell-polynomial likelihood and its AD gradients remain correct regardless of the generator choice. Table 3 uses Clayton exclusively because Clayton admits closed-form Marshall–Olkin sampling; Table 17 repeats the experiment with Frank and Gumbel generators and samples via the Rosenblatt transform. L-BFGS-B recovers all parameters to within 5% at d=10 and within 2.1% at d=20 for both families. The Frank d=10 root parameter shows 13% error because only two sector-level aggregates inform the root; at d=20 with four sectors this error drops below 1%.

Table 16: Replicated recovery on nested Clayton copulas with R=500 Monte Carlo replications per cell. Each replication draws a fresh sample of size N and refits the MLE. EmpSE is the across-replication standard deviation of $\hat{\theta}$; RMSE is its root-mean-squared error against the true value; Bias is the mean of $\hat{\theta} - \theta_{\text{true}}$. HessSE is $\sqrt{\text{diag}((N\hat{H})^{-1})}$ averaged across replications, where $\hat{H} = -N^{-1}\nabla^2\ell(\hat{\theta})$ is the per-observation observed information; every replication produces a positive-definite Hessian. Conv% is the L-BFGS-B convergence rate.

Config	N	Param	True	Mean	EmpSE	HessSE	RMSE	Conv%
2-level	200	θ_{outer}	1.50	1.517	0.112	0.111	0.113	100
d=10		θ_{inner}	3.00	3.023	0.084	0.084	0.087	100
	500	θ_{outer}	1.50	1.521	0.069	0.070	0.072	100
		θ_{inner}	3.00	3.023	0.054	0.054	0.059	100
	1,000	θ_{outer}	1.50	1.519	0.049	0.050	0.053	100
		θ_{inner}	3.00	3.022	0.039	0.038	0.044	100
2-level	200	θ_{outer}	2.00	2.019	0.071	0.074	0.074	100
d=20		θ_{inner}	4.00	4.026	0.077	0.074	0.081	100
	500	θ_{outer}	2.00	2.019	0.048	0.047	0.052	100
		θ_{inner}	4.00	4.023	0.048	0.047	0.054	100
	1,000	θ_{outer}	2.00	2.019	0.035	0.033	0.040	100
		θ_{inner}	4.00	4.027	0.033	0.033	0.043	100
3-level	200	θ_{outer}	1.00	1.019	0.079	0.079	0.081	99.8
d=50		θ_{middle}	2.00	2.019	0.044	0.045	0.048	99.8
		θ_{inner}	4.00	4.024	0.050	0.047	0.056	99.8
	500	θ_{outer}	1.00	1.016	0.052	0.050	0.054	99.4
		θ_{middle}	2.00	2.016	0.029	0.028	0.033	99.4
		θ_{inner}	4.00	4.023	0.032	0.030	0.039	99.4
	1,000	θ_{outer}	1.00	1.018	0.035	0.035	0.039	99.4
		θ_{middle}	2.00	2.019	0.021	0.020	0.028	99.4
		θ_{inner}	4.00	4.023	0.022	0.021	0.032	99.4

Table 17: Parameter recovery on nested Frank and Gumbel copulas (N=500, 2-level nesting, L-BFGS-B optimiser). The experiment samples data via the Rosenblatt transform, then recovers parameters using exact AD gradients.

Family	d	Sectors	Param	True	MLE	Err%
Frank	10	2	θ_{outer}	2.00	1.74	13.0
			θ_{inner}	5.00	4.95	1.1
Gumbel	10	2	θ_{outer}	2.00	1.90	4.9
			θ_{inner}	5.00	5.02	0.3
Frank	20	4	θ_{outer}	2.00	1.98	0.9
			θ_{inner}	4.00	3.94	1.5
Gumbel	20	4	θ_{outer}	2.00	2.04	2.1
			θ_{inner}	4.00	4.04	1.0

N Censoring Validation

The per-variable censoring mechanism reproduces the closed-form mixed partial derivatives of the bivariate and trivariate Clayton and Gumbel CDFs to machine precision across every censoring pattern, and it matches a 50-digit reference on nested trees whose censored leaves sit in subtrees of unequal size. For the Clayton family with generator $\psi(t) = (1+t)^{-1/\theta}$, the k-th order mixed partial of the d-variate CDF admits the closed form

$$\frac{\partial^k C}{\partial u_1 \cdots \partial u_k} = \prod_{j=0}^{k-1}(1+j\theta)\ \prod_{i=1}^{k} u_i^{-(\theta+1)}\ \left(\sum_{i=1}^{d} u_i^{-\theta} - (d-1)\right)^{-(1/\theta+k)}.$$

We compare the output of our Bell-polynomial censoring implementation, which sets $\alpha^{(0)} = [1]$ for censored leaves, against this closed form for $d \in \{2, 3\}$ with every possible censoring pattern running over $k = 0, \ldots, d$ observed variables, 5–6 evaluation points per pattern, and $\theta \in \{0.5, 1, 2, 5\}$. We perform analogous checks for the Gumbel family with $\psi(t) = e^{-t^{1/\theta}}$ and $\theta \in \{1.5, 2, 3, 5\}$, where nested `jax.grad` applied to the known closed-form CDF yields the reference values.

Table 18: Censoring validation: framework output versus closed-form expressions. The test sweep covers every censoring pattern from $k = 0$ to $k = d$ observed variables.

Family	Dimension	Tests	Pass rate	Max $\lvert\Delta \log f\rvert$
Clayton	$d = 2$	48	48/48	3.6×10^{-15}
Clayton	$d = 3$	80	80/80	7.1×10^{-15}
Gumbel	$d = 2$	48	48/48	2.7×10^{-15}
Gumbel	$d = 3$	80	80/80	2.8×10^{-15}

All 256 tests pass with a maximum absolute error of 7.1×10^{-15} in log-likelihood, confirming that the per-variable censoring mechanism correctly reduces to the appropriate mixed partial derivative of a flat copula CDF for every censoring configuration in the sweep.

Flat copulas cannot exercise one step of the nested computation. Before evaluating the density, ACOPULA reorders the children of each node, so a censoring mask supplied per observation must follow the same permutation; the permutation can be non-trivial when sibling subtrees differ in size, and an error in it would flag the wrong leaves as censored. We therefore compare two-level nested Clayton log-densities against a reference that shares no code with ACOPULA: nested numerical differentiation of the closed-form nested CDF with respect to the uncensored leaves only, in `mpmath` at 50 significant digits. Four censored configurations, each at one evaluation point, cover $d \in \{6, 7\}$ with subtree sizes $(3, 4)$, $(2, 4)$, and $(2, 2, 3)$ and one or two censored leaves, at $\theta_{\text{outer}} = 0.8$ and inner parameters between 2.3 and 4.1; a fifth configuration, without censoring, serves as a control. All five agree with the reference to 3.6×10^{-15} in log-density. The same comparison against code that indexes the mask in input order fails all four censored configurations, by 4.6 to 11.9 nats, and passes the control, so the check detects this class of error. The ACOPULA test suite runs these five configurations in `tests/test_nested_censoring_heteropanel.py`.

The five configurations stop at d=7, so we repeat the comparison on the d=53 MIMIC-IV tree of Section 5.3 against the Copulas.jl package of Laverny and Jimenez [2024]. Our port of the nested algorithm into that package, described in Appendix K, shares no code with ACOPULA and computes censored log-densities through the package's generic conditioning interface, which flags each censored leaf by its coordinate index and never reorders the tree. We sample $2{,}000$ observations per family from the fitted Clayton, Gumbel, and Frank trees of Table 4 and censor each lab independently, at per-lab rates that span the cohort's 11.8% to 95.8% and average its 63%. On these observations the two libraries agree to 8.2×10^{-13} in log-density. The code that indexes the mask in input order departs from Copulas.jl on all $1{,}998$ observations per family whose mask mixes censored and observed labs, by 12 to 14 nats at the median, so the comparison detects the same error at full scale. For Gumbel, we evaluate the generator derivatives of Copulas.jl 1.0.1 through its generic Taylor expansion, because its closed form overflows from order 21. The script `experiments/run_copulasjl_crosscheck.sh` reproduces the comparison.

O Supported Archimedean Families

ACOPULA ships with ten Archimedean families—the five classical generators plus five additional generators drawn from Nelsen's catalogue [Nelsen, 2006] that existing software cannot evaluate—and admits arbitrary user-defined generators through a single decorator. Table 19 lists the included generators; the classical families above the rule are standard in the literature, and the five additional families below appear in our experiments. Users can add further families by supplying only the generator function via the `@copula` decorator, after which ACOPULA derives all derivatives, inverses, and sampling procedures automatically.

Table 19: One-parameter Archimedean copula families included in ACOPULA. Classical families appear above the rule; additional families appear below. Sampling uses numerical Laplace inversion for all families.

Family	**Parameter**	**Generator** $\psi_\theta(t)$	**Mixing dist.** F_ψ
Clayton	$\theta > 0$	$(1+t)^{-1/\theta}$	$\mathrm{Gamma}(1/\theta, 1)$
Frank	$\theta > 0$	$-\frac{1}{\theta}\log[1-(1-e^{-\theta})e^{-t}]$	Log-series$(1-e^{-\theta})$
Gumbel	$\theta \geq 1$	$e^{-t^{1/\theta}}$	$S(1/\theta, 1)$ stable
Joe	$\theta \geq 1$	$1-(1-e^{-t})^{1/\theta}$	Sibuya$(1/\theta)$
Ali–Mikhail–Haq	$0 \leq \theta < 1$	$\frac{1-\theta}{e^t-\theta}$	Geometric$(1-\theta)$
InverseGaussian	$\theta > 0$	$e^{(1-\sqrt{1+2\theta t})\,/\,\theta}$	Inv-Gauss$(1,\ 1/\theta)$
Nelsen9	$\theta > 0$	$e^{(1-e^t)\,/\,\theta}$	Poisson$(1/\theta)$
Nelsen12 (Burr)	$\theta \geq 1$	$(1+t^{1/\theta})^{-1}$	Mittag-Leffler$(1/\theta)$
Nelsen13	$\theta \geq 1$	$e^{1-(1+t)^{1/\theta}}$	—
Nelsen17	$\theta > 0$	$\left[1+(2^{-\theta}-1)e^{-t}\right]^{-1/\theta}-1$	—

P High-Dimensional Nested Scaling

This appendix establishes that the $O(d^2)$ complexity bound of Section 3.2 holds in practice across three orders of magnitude in d, and that the prefactor stays small enough for single-observation densities to evaluate in milliseconds at d=1,000. We measure single-observation log-likelihood wall-clock for nested Clayton copulas at $d \in [10, 8{,}000]$ on a single AMD Threadripper 3960X core under float64. We scan three topologies: *fixed* K=5 two-level (root over M=$d/5$ Clayton sectors of five leaves) and three-level (mid layer of five sectors per group); and *balanced* $\sqrt{d}$ (M=K=$\lceil\sqrt{d}\,\rceil$). All times are medians over five post-warmup calls of a single JIT-compiled XLA program; Figure 2 plots the curves and Table 20 lists selected dimensions.

Two regimes. The empirical scaling exhibits two regimes separated by d≈1,000: a sub-quadratic overhead-dominated regime at the dimensions our experiments use, and the predicted $O(d^2)$ asymptote past d≈4,000 (Figure 2). In the *real-data regime* that covers our experiments—retinopathy at d=2, MIMIC-IV at d=53, and S&P 500 at d=98—the empirical exponent is sub-linear, $\alpha \approx 0.55$ on $d \in [10, 100]$. The Bell-polynomial work scales as $c \cdot d^2$ with $c \approx 10^{-5}$ ms, so per-call XLA dispatch and the per-sector `lax.scan` that walks the tree dominate the small quadratic constant. Across the measured regime $d \in [100, 8{,}000]$ a global OLS log-log fit gives $\alpha \approx 2.03$ at 2-level and $\alpha \approx 1.92$ at 3-level, both consistent with the theoretical $O(d^2)$ bound of Section 3.2. The local slopes in Table 20 show a more nuanced picture: the 2-level slope reaches α_Δ=2.06 at d=8,000, settled at the asymptote, while the 3-level slope at d=8,000 is α_Δ=2.89, super-quadratic. This last value reflects L2 cache pressure as the per-sector polynomial powering exceeds cache capacity, not an algorithmic shift; the pure-Cauchy microbenchmark of Appendix A.7 isolates the cache transition from the algorithmic complexity. The balanced $\sqrt{d}$ topology hits the cache wall earlier because each sector contains K=$\lceil\sqrt{d}\,\rceil$ leaves—about 80 at d=6,400—inflating its $d \geq 1{,}000$ slope to $\alpha \approx 3.24$.

Head-to-head against R `nacLL`: balanced $\sqrt{d}$ topology. R's runs exceed our 900-second harness cap from d≈40 under both the fixed-K=5 topology of Figure 2 and the harder balanced

$M{=}K{=}\lceil\sqrt{d}\,\rceil$ topology, where the per-sector arity itself grows with d (Figure 3). The dominant per-density cost in R `nacLL` is the partition enumeration in `b.coeff` from the `copula` source file `R/Rsource/dnac.R`, lines 95–145: for the Hofert–Pham formula, the inner loop is $O(n \cdot d_0 \cdot \prod_v K_v)$, where d_0 is the root arity and K_v are the per-sector sizes; with balanced sectors of size K and $M{=}d/K$ sectors this becomes $O(K^M)$ per density, exponential in the number of sectors. The classical $O(d^2)$ Stirling-table closed form of Hofert et al. [2013] applies only to flat single-level Archimedean copulas; for nested copulas no Stirling shortcut exists once $M \geq 2$. Timing R at $n{=}1$ requires a two-line patch to `nacLL` that adds `drop=FALSE` at `dnac.R` line 189; upstream `nacLL` only handles $n{\geq}2$.

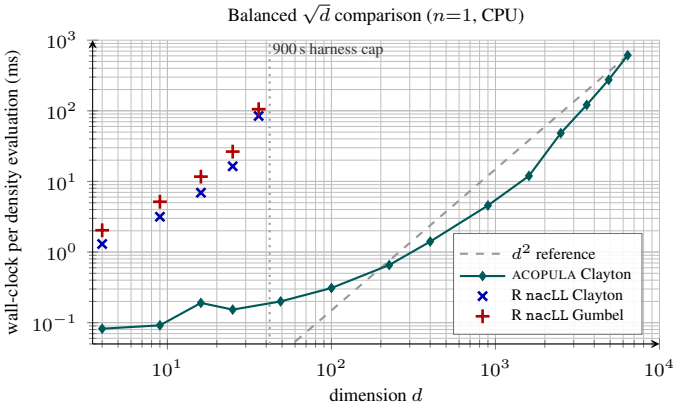


Figure 3: Comparison with R `nacLL` under the balanced $M{=}K{=}\lceil\sqrt{d}\,\rceil$ topology, $n{=}1$ single-evaluation timings, single-thread CPU, float64. From $d{\approx}40$, R's Clayton/Gumbel runs exceed our 900-second harness cap; R errors on Frank for every d. ACOPULA extends to $d{=}6{,}400$ with the same $O(d^2)$ asymptote as the fixed-$K{=}5$ topology (Figure 2); the slope steepening in $d \in [1{,}600,\ 6{,}400]$ is a hardware cache transition as each sector size $K{=}\lceil\sqrt{d}\,\rceil$ approaches 80 leaves (Appendix A.7).

Table 20: Single-observation wall-clock at selected dimensions on AMD Threadripper 3960X (single thread, float64). α_Δ is the local log-log slope from the previous row, illustrating the transition from overhead-dominated ($\alpha_\Delta \ll 2$) at small d to the $O(d^2)$ asymptote ($\alpha_\Delta \approx 2$) at large d; intermediate slopes also reflect L1/L2 cache transitions, not algorithmic shifts.

	2-level (fixed $K{=}5$)			3-level (fixed $K{=}5$)		
d	JIT (s)	Exec (ms)	α_Δ	JIT (s)	Exec (ms)	α_Δ
10	0.78	0.10	—	1.31	0.12	—
100	0.78	0.28	0.45	1.12	0.35	0.47
1,000	1.59	16.16	1.76	1.68	7.38	1.32
4,000	3.47	150.55	1.61	2.61	195.81	2.36
8,000	6.40	629.91	2.06	5.53	1454.98	2.89

Q Memory and Compilation Benchmarks

The compilation cost a user pays once at fit time grows linearly in d, the gradient program stays at the standard reverse-mode $\sim 3\times$ multiple of the forward program, and the entire fused HLO fits in a few thousand instructions—small enough that the JIT cache amortises across the inner-loop evaluations of any optimiser. Table 21 reports compilation overhead, execution time, and XLA program size for the Bell polynomial likelihood and its gradient at increasing dimensions d. All measurements use a two-level nested Clayton copula at sector size $K{=}5$, on an AMD Threadripper 3960X CPU (3.8 GHz, 24 cores / 48 threads) with JAX 0.8.3 in float64 mode. We count HLO instructions in the single fused program XLA emits via `jax.jit(f).lower(x).compile().as_text()`.

Three patterns stand out from Table 21. First, JIT compilation time stays nearly flat in d because the stage-group batched likelihood compiles each of the two nesting levels into one fused `lax.scan`;

Table 21: Compilation and execution benchmarks for the Bell polynomial likelihood and gradient on CPU (AMD Threadripper 3960X, n=1 observation). "JIT" denotes one-time compilation time; "Exec" denotes the median execution time over 10 post-warmup runs. "HLO ops" counts instructions in the compiled XLA program.

	JIT compile (s)		Execution (ms)		HLO ops	
d	Fwd	Grad	Fwd	Grad	Fwd	Grad
10	1.09	1.85	0.092	0.291	1 707	4 981
20	0.83	1.62	0.099	0.216	1 741	5 180
50	0.73	1.45	0.112	0.914	1 782	5 342
100	0.75	1.81	0.341	1.061	1 778	5 344

growing d extends the scan length but not the compiled program. Second, the compiled HLO program grows minimally—from 1 700 to 1 800 forward ops and 5 000 to 5 300 gradient ops—confirming that `lax.scan` reuses the same loop body rather than unrolling. Third, the gradient program contains roughly $3\times$ the forward operations, matching the standard reverse-mode AD multiple for scan-based programs.

R Exact MLE vs. Simulated Maximum Likelihood

Exact MLE through our Bell-polynomial likelihood dominates the *simulated maximum likelihood estimation* (SMLE) approach that R `nacopula` [Hofert et al., 2013] relies on, in both parameter accuracy and wall time, at every SMLE sample budget we test. The SMLE estimator draws N frailty samples from the Laplace-transform hierarchy at each parameter evaluation, computes the conditional density at each observation given those frailties, and averages over the N draws to approximate the intractable marginal density. Because the outer copula parameter enters only through the frailty distribution—not through the conditional density—the SMLE objective demands fresh frailty samples at every evaluation, producing a noisy objective that forces derivative-free optimisation. Our framework instead computes the exact density and exact gradients, opening the door to L-BFGS-B with analytic gradients.

Setup. We sample 500 observations from a two-level nested Clayton copula at d=10 with two sectors of size K=5, θ_{outer}=1.5, and θ_{inner}=3.0. For each of 30 replications, we fit four estimators on the same data: exact MLE via our Bell-polynomial likelihood, and SMLE at $N \in \{100,\ 1{,}000,\ 10{,}000\}$ frailty samples per evaluation. The exact MLE uses L-BFGS-B with analytic gradients; SMLE uses differential evolution—a stochastic global optimiser for noisy objectives—with frailty samples re-drawn each evaluation.

Table 22: Exact MLE vs. SMLE on simulated nested Clayton (d=10, sample size n=500, 30 replications; N denotes the number of frailty samples per SMLE evaluation, distinct from the sample size n). True parameters: θ_{outer}=1.5, θ_{inner}=3.0. RMSE_{o}, RMSE_{i} denote the root-mean-squared error for $\hat{\theta}_{\text{outer}}$ and $\hat{\theta}_{\text{inner}}$ respectively.

Method	$\hat{\theta}_{\text{outer}}$ (mean±sd)	RMSE_{o}	$\hat{\theta}_{\text{inner}}$ (mean±sd)	RMSE_{i}	Wall (s)
Exact MLE	$\mathbf{1.527 \pm 0.064}$	$\mathbf{0.069}$	$\mathbf{3.016 \pm 0.060}$	$\mathbf{0.061}$	$\mathbf{1.5}$
SMLE N=100	1.414 ± 0.231	0.243	2.638 ± 0.184	0.405	5.9
SMLE N=1,000	1.413 ± 0.185	0.202	2.939 ± 0.127	0.139	16.7
SMLE N=10,000	1.516 ± 0.123	0.122	2.982 ± 0.073	0.074	153.1

Table 22 reports the results. The exact MLE recovers both parameters with near-zero bias, RMSE ≈ 0.07 on each, in 1.5 s per fit. SMLE at N=100 underestimates θ_{inner} by 0.36 on average, with an RMSE of 0.40—roughly six times the exact MLE's error at four times the wall time. Increasing N to 10,000 reduces the bias to below 0.02 and brings the RMSE to 0.07 on θ_{inner}, matching the exact MLE's accuracy—but at 153 s per fit, *one hundred times slower* than exact inference on the same data. For the moderate dimensions typical of actuarial and survival-analysis applications, $d \leq 100$, the exact approach dominates SMLE in both accuracy and wall time at every SMLE sample budget

tested. Differentiability compounds the advantage: L-BFGS-B with analytic gradients converges in far fewer iterations than the derivative-free global search SMLE's noisy objective requires.

Generator-derivative Monte Carlo: Gen-AC ablation. The original Gen-AC training procedure [Ng et al., 2021] replaces $\psi^{(k)}(t) = \mathbb{E}_U[(-e^{\mathrm{MLP}(U)})^k \exp(-e^{\mathrm{MLP}(U)}t)]$ with an N-sample Monte Carlo average $\hat{\psi}_N^{(k)}$, mirroring SMLE's role above but at the generator-derivative level rather than the joint-likelihood level. The resulting log-likelihood is biased by Jensen's inequality on $\log$ composed with the nonlinear Hofert density formula in $\hat{\psi}^{(k)}$, with bias $O(1/N)$—biased but consistent. We measure this bias on Gen-AC-Fourier with $K{=}2$ Fourier features, hidden width 3, 262 parameters, MIMIC-IV $d{=}53$, $N{=}32$ frailty samples per evaluation, seed 42. Table 23 reports a four-cell ablation: each of two trained models (θ_{stoch} trained with re-sampled Monte Carlo gradients, θ_{quad} trained with deterministic quadrature gradients) evaluated under both estimators (the stochastic estimator averages 50 independent draws to estimate $\mathbb{E}[\text{stoch NLL}]$). Eval-time Jensen plus correlation bias accounts for 0.03–0.07 nats; train-time optimisation differences contribute about 0.03 nats; the quadrature-trained, quadrature-evaluated model attains the lowest NLL.

Table 23: Gen-AC-Fourier$_{260}$ on MIMIC-IV ($d{=}53$): held-out NLL under each combination of training and evaluation estimator, seed 42. Lower is better. Rows index the trained model, columns the evaluation method; θ_{stoch} trains with $N{=}32$ Monte Carlo frailty draws per Adam step, θ_{quad} trains with $N{=}32$ Gauss–Legendre quadrature nodes per Adam step.

	quad eval (truth)	stoch eval (50-MC avg)
θ_{stoch}	7.131	7.165
θ_{quad}	**7.096**	7.166

S Predictive Validation

We report held-out predictive log-likelihoods for the two real-data experiments, MIMIC-IV at $d{=}53$ and the diabetic retinopathy paired-eye study at $d{=}2$. For MIMIC-IV, we verify that the in-sample family ranking of Section 5.3 survives an 80/20 train/test split. For retinopathy, an 80/20 split checks whether any family overfits in sample, and Appendix S.2.2 then measures how much of the resulting ordering 197 patients can resolve at all.

S.1 MIMIC-IV

The MIMIC-IV ablations break into three pieces: a held-out evaluation that confirms the in-sample family ranking, a compile-cost audit that quantifies the one-time XLA price of the Frank generator, and a Hessian-based standard-error check that confirms parameter identifiability at $n_{\mathrm{train}}{=}68{,}183$.

Cohort construction. The cohort draws from MIMIC-IV 3.1 on PhysioNet: one row per `hadm_id` with an ICU stay, joined on `charttime` to lab events that fall inside the ICU window `[intime, outtime]`. For each admission-lab pair, we record T_{ij}, the hours from ICU intime to the first abnormal reading of lab j, and right-censor at the ICU length-of-stay when no abnormal reading occurs during the window. The "abnormal" label uses MIMIC's native `flag='abnormal'` column, which MIMIC curators set against each lab's own reference range; we adopt this labelling verbatim and apply no additional reference range. A lab enters the cohort only when at least 30% of admissions have a measurement, at least 5% of admissions have an abnormal reading, and at least 90% of its rows carry a numeric `valuenum`. These thresholds select the $d{=}53$ itemids in Table 24, grouped into six named lab panels plus one catch-all following the clinical groupings hard-coded in `experiments/shared/mimic.py`. Running `experiments/data/build_mimic.py` on the MIMIC-IV 3.1 raw CSVs materialises the cohort as an ($n{=}85{,}229$, $d{=}53$) censored matrix in `mimic_full_censored.npz`.

Censoring assumption. The two-stage fit assumes that each admission's censoring time is independent of its lab event times, an assumption we adopt to keep the illustration simple. A single administrative time censors all 53 labs of an admission: each of the 2,856,159 censored cells equals the admission's ICU length of stay. The assumption is strong for this cohort, because illness severity

can both lengthen the stay and hasten abnormal labs. We assess the assumption on the $82{,}447$ admissions that stay at least 12 hours and carry a censored lab, 96.7% of the cohort. No censoring acts on these admissions before 12 hours, yet the mean number of labs already abnormal at 12 hours rises across length-of-stay quartiles from 11.24 to 16.17. Per lab, a pooled two-proportion z-test of the top quartile against the bottom one rejects equality for all 53 labs at the Bonferroni level $0.05/53$, every one in the same direction. A violation biases the Kaplan–Meier marginals and the fitted association parameters of Section 5.3, including the per-panel Kendall's τ. The density evaluation never refers to the censoring mechanism, and the S&P 500 data carry no censoring, so the assumption does not enter the correctness validation of Appendix N, the complexity results, or the S&P 500 fit. Under informative censoring, the censoring time may join the copula as one more margin, as Li et al. [2023] do for two event times by placing death in the outer layer. Algorithm 1 evaluates that likelihood for any tree; we leave the fit to a separate study.

Table 24: MIMIC-IV 3.1 cohort panels used in the nested copula fit. Itemids come from MIMIC's `d_labitems` dictionary; the "Other" panel collects the 14 labs that pass the quality filters but fall outside the six named groupings, with the build-time pipeline fixing its membership.

Panel	Count	`itemids`
CBC	10	$51221, 51265, 51222, 51277, 51279, 51301, 51249, 51250, 51248, 52172$
CMP	11	$50902, 51006, 50983, 50971, 50912, 50882, 50868, 50931, 50960, 50893, 50970$
Diff	4	$51254, 51200, 51256, 51244$
Liver	6	$50885, 50878, 50861, 50863, 50862, 50954$
Coag	3	$51237, 51274, 51275$
ABG	5	$50813, 50820, 50821, 50818, 50804$
Other	14	determined at build time from the quality filter
Total	**53**	

Held-out evaluation. We repeat the held-out protocol on the MIMIC-IV ICU lab dataset with $d{=}53$ labs in a seven-panel nested hierarchy, using an 80/20 admission-level split at seed 42. Stage 1 estimates each lab's marginal survivor function with Kaplan–Meier on training data only; Stage 2 fits copula parameters on the training pseudo-observations via L-BFGS-B with analytic gradients. Table 25 reports per-observation NLL on both splits ($n_{\text{train}}{=}68{,}183$, $n_{\text{test}}{=}17{,}046$), alongside the mean over the five re-randomised splits of Appendix S.1.2 and the paired gap that prices the choice of split.

Table 25: MIMIC-IV: held-out predictive NLL per observation. The symbol $-\bar{\ell}$ denotes the per-observation negative log-likelihood, so lower is better in every numeric column, and the mean column averages it over the five re-randomised $80/20$ splits of Appendix S.1.2. The column Δ holds the across-split paired gap to the pre-specified reference Clayton/Clayton in nats per admission, so a Δ below zero would mark a nesting that beats it, with the Nadeau–Bengio interval that prices the 80% train-set overlap between splits [Nadeau and Bengio, 2003]. Neither interval covers zero, and both gaps clear a Holm correction over the three comparisons and the simultaneous band alike, so both nestings separate from Clayton/Clayton. $n_{\text{train}}{=}68{,}183$, $d{=}53$, 8 copula parameters.

	published split, seed 42				across the 5 splits	
Nesting	$-\ell_{\text{train}}$	$-\ell_{\text{test}}$	$-\bar{\ell}_{\text{train}}$	$-\bar{\ell}_{\text{test}}$	mean $-\bar{\ell}_{\text{test}}$	Δ [95% CI]
Clayton/Clayton	481,974.8	120,817.6	7.069	7.088	7.034	reference
Frank/Frank	504,459.2	126,849.7	7.399	7.442	7.373	0.339 $[0.304, 0.373]$
Gumbel/Gumbel	544,413.4	137,087.4	7.985	8.042	7.970	0.936 $[0.893, 0.979]$

The held-out ranking matches the in-sample AIC ranking of Table 26: Clayton/Clayton attains the lowest test NLL, with a small train-to-test gap (7.069 vs 7.088) thanks to the fully non-parametric Kaplan–Meier marginals that avoid the per-lab tail-misspecification bias. Both Frank/Frank and Gumbel/Gumbel fit worse than Clayton at each of the five splits, and their across-split intervals stay clear of zero by more than 0.30 nats. Table 27 reports one-time fresh-JIT costs on an RTX 3090 Ti for the MIMIC-IV dynamic-censoring pipeline at $n_{\text{train}}{=}68{,}183$. `ACOPULA.set_compile_cache_dir` caches every compile to disk, so subsequent runs read the compiled HLO program back in under a second.

Table 26: MIMIC-IV in-sample model selection on the seed-42 split, all eleven generators of Table 4 in that table's order. We define AIC$/n$ $= 2(-\bar{\ell}_{\text{train}}) + 2|\theta|/n$ with n=68,183 training admissions, so every entry follows from the two columns beside it, up to the rounding of $-\bar{\ell}_{\text{train}}$, which moves AIC$/n$ by at most 0.0014. Lower is better in $-\bar{\ell}_{\text{train}}$ and in AIC$/n$. Table 4 carries the held-out counterpart.

Family	$\lvert\theta\rvert$	$-\bar{\ell}_{\text{train}}$	AIC$/n$
Frank/Frank	8	7.399	14.797
Gumbel/Gumbel	8	7.985	15.969
Clayton/Clayton	8	7.069	14.138
AMH/Clayton‡	8	7.441	14.882
ACNet$_{(4,4)/\text{I4}}$	70	7.058	14.119
ACNet$_{(8,8)/\text{I4}}$	130	7.058	14.120
ACNet$_{(4,4,4)/\text{I4}}$	90	7.058	14.120
Gen-AC$_{1.1\text{k}}$	1,142	7.228	14.489
Gen-AC-Fourier$_{260}$	260	7.141	14.290
Gen-AC-Fourier-$\int_{260}$	260	7.069	14.146
Gen-AC-Fourier$_{3.6\text{k}}$	3,622	7.084	14.275

Table 27: MIMIC-IV: one-time XLA compile wall time on an RTX 3090 Ti. The `val+grad` column times JIT compilation of `jax.value_and_grad` on the chunked negative log-likelihood; the `Hessian compute` column times the chunked Hessian pass that supplies observed Fisher standard errors.

Family	val+grad JIT	Hessian compute
Clayton/Clayton	16 s	25 s
Gumbel/Gumbel	16 s	26 s
Frank/Frank	25 s	44 s

A warm cache reproduces the full Table 28 in under five minutes (L-BFGS-B fit plus chunked Hessian compute on already-compiled kernels).

Standard errors. `jax.hessian` on the MIMIC-IV fits yields positive-definite Hessians and tight standard errors for all three nestings at n_{train}=68,183; see Table 28. Every coefficient of variation (CV) stays below 1%, so the L-BFGS-B optima are sharply identified.

Table 28: MIMIC-IV: Hessian-based standard errors across three nestings, computed by `jax.hessian` with a delta-method projection to the constrained space. All three Hessians are positive definite, and the coefficient of variation stays below 1% for every parameter. n_{train}=68,183.

	Clayton/Clayton		Gumbel/Gumbel		Frank/Frank	
Parameter	$\hat{\theta}$	CV (%)	$\hat{\theta}$	CV (%)	$\hat{\theta}$	CV (%)
θ_{outer}	0.3105	0.5	1.1203	0.1	1.1618	0.5
θ_{CBC}	0.6031	0.5	1.2266	0.1	2.0998	0.4
θ_{CMP}	0.4680	0.5	1.1887	0.1	1.7454	0.4
θ_{Diff}	4.2698	0.7	1.7292	0.3	7.7220	0.5
θ_{Liver}	2.1079	0.7	1.4149	0.2	4.3909	0.5
θ_{Coag}	2.3719	0.6	1.7712	0.3	5.7236	0.5
θ_{ABG}	2.3887	0.5	1.5964	0.2	5.3496	0.4
θ_{Other}	1.2368	0.7	1.2618	0.1	3.0185	0.5

S.1.1 Ablation: independence and single-layer baselines

To quantify the contribution of the hierarchical nesting structure, we evaluate two simpler alternatives on the same held-out split: an independence copula whose mixed partial of $C(\mathbf{u})=\prod_j u_j$ in the uncensored variables collapses to $\prod_{j:\,\delta_j=0} u_j$, contributing $\sum_{j:\,\delta_j=0} \log u_j < 0$ per observation; and

a single-layer copula that places every lab under a single root node with one parameter, ignoring the panel grouping. Table 29 reports the results. The hierarchical structure captures intra-panel dependence that a single exchangeable copula cannot represent. The eight-parameter nested Clayton improves test NLL substantially over the best flat copula, because the panel structure absorbs heterogeneity across the 53 labs into per-panel parameters instead of forcing a single exchangeable parameter to average over the cohort.

Table 29: MIMIC-IV ablation: independence and single-layer baselines versus the best nested model on the held-out split (n_{train}=68,183, d=53, 8 copula parameters, Kaplan–Meier marginals throughout). Flat copulas use a single exchangeable parameter regardless of dimension.

Model	#Params	$-\bar{\ell}_{\text{train}}$	$-\bar{\ell}_{\text{test}}$
Independence	0	12.340	12.344
Flat Clayton	1	9.204	9.193
Flat Gumbel	1	9.746	9.751
Nested Clayton/Clayton	8	7.069	**7.088**

S.1.2 Uncertainty of the MIMIC-IV family ranking

Several rows of Table 4 differ only in the third decimal, so we measure how much of that ordering 85,229 admissions resolve. We re-randomise the $80/20$ admission-level split at five seeds, 42 through 46, and on each split we refit every classical combination and every neural configuration from scratch, Kaplan–Meier marginals included, then score it on that split's held-out admissions: 20 classical fits and 35 neural fits, at n_{train}=68,183 and n_{test}=17,046 throughout. Every fit converges. The published seed-42 split is one of the five, and it is the hardest of the five: all eleven models score their worst held-out likelihood there, Clayton/Clayton at 7.088 against a five-split mean of 7.034. The neural gaps that Table 4 prints come from this refit at seed 42, which moves each neural level by up to 0.001 nats from the published fit printed beside it, so a gap need not equal the difference of the two printed levels.

Pre-specified reference. Every interval below compares one model against Clayton/Clayton, and the copula NLL on the split's own *training* admissions chooses that reference, never the held-out scores. The rule returns Clayton/Clayton independently at all five splits. Choosing the reference by its held-out score and then reading intervals off those same admissions is test-set reuse, and Appendix S.2.2 measures what the reuse costs on the smallest geometry in this paper, ten families over 197 and over 40 patients: the step-down rule holds its nominal 5% family-wise error—the chance of at least one false rejection anywhere in the table—at 3.3% and 6.0% under a pre-specified reference, and drifts to 15.3% and 16.7% once the held-out scores pick it.

Two replicate axes, two interval estimators. Within one split the held-out admissions are the replicate. We resample the 17,046 of them B=10,000 times from a single shared index stream, so every model sees the same resampled admissions and every interval pairs a model against the reference admission by admission. With the reference pre-specified, the headline decision on that axis is the Romano–Wolf step-down p-value [Romano and Wolf, 2005], which resamples the largest *studentised* statistic, an estimate divided by its own standard error; the Holm-adjusted p-value [Holm, 1979] and the all-pairs studentised band accompany it as the conservative reads. Across splits the partition is the replicate, and the naive standard error of five split means ignores that any two training sets share 80% of their rows. The Nadeau–Bengio correction [Nadeau and Bengio, 2003] replaces the naive $1/J$ by $1/J + n_{\text{test}}/n_{\text{train}}$, here $0.200 + 0.250$, and refers the result to a Student-t on four degrees of freedom, so it widens every standard error by exactly 1.5 and is algebraically exact for an $80/20$ partition. Holm over the comparisons against the reference is the headline on that axis, with a studentised max-t simultaneous band drawn from one shared bootstrap stream over the five split rows as its companion. One uncorrected Nadeau–Bengio interval per model is not a 5% statement: on this geometry seven such intervals carry a family-wise error near 10% under an exact null, which Holm returns to about 1%.

What the across-split spread measures. Each replicate re-deals the same 85,229 admissions into a training and a test half, so the spread over the five splits is partition noise on a fixed cohort and

not the sampling variability of a fresh cohort of ICU admissions. Each split does score a disjoint random 20%, so the observation-sampling component is 80% present rather than absent, and the resulting understatement is exactly $\sqrt{1 - n_{\text{test}}/n_{\text{cohort}}} = 0.894$: the standard deviation we report is about 11% smaller than the same statistic would show over fresh cohorts of this size. A pooled cross-validation design, where every repeat rescores the whole cohort, drops that component entirely and understates by a much larger factor; the retinopathy study of Appendix S.2.2 has that geometry, and its factor does not carry over to this one. The quantity p_{best} in Table 30 is likewise the fraction of partitions a model wins on this cohort and not a probability of being best, and it degenerates to a $0/1$ indicator as partitions accumulate. The Nadeau–Bengio interval is what prices this geometry.

The classical families separate. Clayton/Clayton takes the lowest held-out NLL at every one of the five splits, and Frank/Frank, AMH/Clayton, and Gumbel/Gumbel hold ranks two, three, and four at every split without exception. The across-split gaps are 0.339 nats per admission for Frank/Frank, 0.371 for AMH/Clayton at a five-split mean of 7.405, and 0.936 for Gumbel/Gumbel, at t statistics of 27.2, 42.3, and 60.3 on four degrees of freedom. All three clear Holm and the simultaneous band alike, so three of three comparisons separate under both rules, and Table 25 carries the two same-family intervals.

The Clayton-versus-ACNet ordering does not resolve. On the published split the three ACNet configurations sit 0.004 nats behind Clayton/Clayton, every interval covers zero, and the smallest step-down p-value is 0.416. Averaged over the five splits the same three sit 0.005 nats ahead of it, every Nadeau–Bengio interval covers zero, and the Holm-adjusted p is 0.832 for all three. The sign of the gap therefore follows the split: Clayton/Clayton leads at the published split alone and trails at the other four, its mean rank over the five is 3.4 across a range of 1 to 4, and ACNet$_{(4,4,4)/\text{I4}}$ wins three of the five outright. No ACNet configuration separates from Clayton/Clayton at any single split either. The ordering between the two is unresolvable at this cohort size, and the experiment supports neither a claim that ACNet beats Clayton/Clayton nor the reverse.

The weaker Gen-AC configurations separate. Gen-AC$_{1.1\text{k}}$ trails Clayton/Clayton by 0.142 nats across the five splits, with an interval of $[0.108,\ 0.177]$ and a Holm-adjusted p of 0.002, and Gen-AC-Fourier$_{260}$ trails by 0.089 at p=0.013; both also separate at each of the five splits taken singly. The remaining two configurations show what pricing the split costs. Gen-AC-Fourier$_{3.6\text{k}}$ separates from Clayton/Clayton within all five splits individually and clears the step-down at the published split at p=0.027, yet its gap moves from 0.014 to 0.089 nats between splits, so the across-split interval $[-0.007,\ 0.098]$ covers zero at a Holm-adjusted p of 0.370. That same row fails the two conservative within-split rules at the published split as well, at a Holm-adjusted p of 0.058 and inside the all-pairs band. Gen-AC-Fourier-$\int_{260}$ separates at three of the five splits and not across them, at 0.020 and p=0.370.

Two noise scales. The split-to-split standard deviation of the held-out NLL is 0.037 for Clayton/Clayton and 0.040 for each ACNet configuration, while the gap between them is 0.004 to 0.005, roughly eight times smaller than the spread. The relation between the two numbers is what makes them readable together: a partition that deals hard admissions into the test half moves every model at once, so that shared component cancels when the comparison pairs two models admission by admission, which is why the paired standard error is 0.005 rather than 0.037. The gain belongs to the pair compared and not to the procedure, and it varies: the ratio of the marginal to the paired standard error runs from 7.11 to 11.90 over the neural comparisons at the published split and from 2.52 to 5.89 over the classical ones, against 1.56 to 16.27 over the eight comparisons to the reference on the S&P 500 panel of Appendix W.2, where folds rather than test days supply the resampling unit and pairing helps throughout.

Initialisation noise. Each cell of the five-split study carries one fresh initialisation, so its across-split spread confounds partition noise with optimisation noise, and we separate the optimisation component at the published split. We refit ACNet$_{(4,4)/\text{I4}}$ at five independent random initialisations, `init_key` 0 through 4, on one train/validation/test partition under the recipe of Table 4: $10{,}000$ Adam steps with a 500-step warmup, cosine learning-rate decay from a peak of 10^{-2}, early-stop patience 2000, a 10% validation split, and chunk size 512. Across the five fits the test NLL per observation is 7.0920 with a standard deviation of 0.001 and a range of 7.091 to 7.094, an order of magnitude below the 0.037–0.040 partition spread, so the partition dominates the optimiser. For each fit we also form

Table 30: MIMIC-IV neural generators against the pre-specified classical reference over five re-randomised $80/20$ splits of the fixed $85{,}229$-admission cohort, seeds 42 through 46, with every model refit from scratch on each split. The left group summarises the five per-split held-out NLLs per admission, where the rank range gives the best and worst rank a model attains over those splits and p_{best} the fraction of splits it wins; those five columns describe one fixed cohort under different partitions and none of them is sampling variability across cohorts. The right group holds the across-split paired gap to Clayton/Clayton in nats per admission, its Nadeau–Bengio interval [Nadeau and Bengio, 2003], and the Holm-adjusted p-value over the seven comparisons. The mark $\approx$ reads the unadjusted interval on its own row and flags the five gaps that cover zero; the Holm column corrects for the seven comparisons at once and agrees with it row for row. Only the two weakest Gen-AC rows separate; the three ACNet rows lead on the mean and on the rank while sitting well inside their intervals.

	across the 5 splits					gap to Clayton/Clayton, $\Delta<0$ better		
Model	mean $-\bar{\ell}_{\text{test}}\downarrow$	sd	mean rank	rank range	p_{best}	Δ [95% CI]	tie	p_{Holm}
ACNet$_{(4,4,4)/\text{I4}}$	7.029	0.040	1.4	1–2	0.60	-0.005 $[-0.017, 0.006]$	$\approx$	0.832
ACNet$_{(8,8)/\text{I4}}$	7.029	0.040	2.2	1–3	0.20	-0.005 $[-0.017, 0.007]$	$\approx$	0.832
ACNet$_{(4,4)/\text{I4}}$	7.030	0.040	3.0	2–4	0.00	-0.005 $[-0.016, 0.007]$	$\approx$	0.832
Clayton/Clayton	7.034	0.037	3.4	1–4	0.20	reference		
Gen-AC-Fourier-$\int_{260}$	7.054	0.036	5.2	5–6	0.00	0.020 $[-0.004, 0.044]$	$\approx$	0.370
Gen-AC-Fourier$_{3.6\text{k}}$	7.080	0.022	5.8	5–6	0.00	0.045 $[-0.007, 0.098]$	$\approx$	0.370
Gen-AC-Fourier$_{260}$	7.123	0.030	7.0	7–7	0.00	0.089 $[\ 0.054, 0.124]$		0.013
Gen-AC$_{1.1\text{k}}$	7.177	0.052	8.0	8–8	0.00	0.142 $[\ 0.108, 0.177]$		0.002

B=10,000 paired bootstrap resamples of $\bar{\Delta} = (-\bar{\ell}^{\text{ACNet}}_{\text{test}}) - (-\bar{\ell}^{\text{Clayton}}_{\text{test}})$ over the 17,046 held-out admissions, the same difference of negative log-likelihoods that every other Δ in this paper measures, and Table 31 reports the per-fit interval together with the bootstrap probability $\Pr(\bar{\Delta} > 0)$ that Clayton/Clayton wins the resampled mean. Every interval covers zero and Clayton's win rate sits between 74% and 87%, a directional preference that reaches significance at no initialisation.

Table 31: Paired bootstrap on $\bar{\Delta} = (-\bar{\ell}^{\text{ACNet}}_{\text{test}}) - (-\bar{\ell}^{\text{Clayton}}_{\text{test}})$ across B=10,000 resamples of the $17{,}046$ test admissions, per ACNet initialisation, at the published seed-42 split. The bar averages over the resampled admissions, and $\Pr(\bar{\Delta}>0)$ scores that average, so a $\bar{\Delta}$ below zero would mark an initialisation that beats Clayton/Clayton. Every 95% CI covers zero, so no initialisation separates ACNet from Clayton/Clayton.

	gap to Clayton/Clayton	
`init_key`	$\bar{\Delta}$ [95% CI]	$\Pr(\bar{\Delta}>0)$
0	0.003 $[-0.007, 0.013]$	0.74
1	0.004 $[-0.006, 0.013]$	0.77
2	0.004 $[-0.006, 0.014]$	0.79
3	0.006 $[-0.004, 0.015]$	0.87
4	0.005 $[-0.006, 0.014]$	0.83

Cross-driver alignment. Two drivers produce this study, one for the classical combinations and one for the neural configurations, and both must score the same admissions in the same order or the pairing dissolves without warning. Every cell therefore records the SHA-256 hash of its test-row index vector, and the drivers compare those hashes rather than row counts, which sit at $17{,}046$ on every split and so separate nothing. On the shared Clayton/Clayton per-admission vector the two agree exactly, to the last bit: over the full held-out vector at the published split, and over a deterministically sampled 512-row re-scoring at each of the other four.

What the experiment supports. The cohort separates the three remaining classical families from Clayton/Clayton decisively, and it separates the two weakest Gen-AC configurations from it as well, while it separates no ACNet configuration from it in either direction. The third-decimal ordering among Clayton/Clayton and the three ACNet rows is a property of the partition.

S.1.3 Marginal misspecification

We refit each of the three classical pairs under two parametric marginals—per-lab Weibull and per-lab lognormal fitted by maximum likelihood with right-censoring—in addition to the Kaplan–Meier marginals of the body, on the full cohort with n=85,229. Table 32 reports per-observation copula NLL and the fitted root parameter $\hat{\theta}_{\text{outer}}$, and two findings emerge. First, the *ranking* of copula families is stable across all three marginals: Clayton/Clayton attains the lowest copula NLL under Kaplan–Meier, Weibull, and lognormal alike, with Gumbel and Frank trailing in every case. Second, the fitted *parameters* shift with the marginal by family: Gumbel is essentially invariant ($\hat{\theta}_{\text{outer}} \in [1.116, 1.122]$, per-panel inner parameters within 8%), whereas Clayton moves most ($\hat{\theta}_{\text{outer}}$ from 0.21 to 0.31, a 32% spread, and up to 38% on individual panels) and Frank lies between (20% outer, 14% inner). The family ranking is therefore robust to the marginal model, but the magnitude of the dependence parameters—especially for Clayton—should be read together with the marginal under which it was fitted.

Table 32: MIMIC-IV marginal sensitivity (n=85,229, d=53, full cohort, 8 copula parameters per fit). Per-observation copula NLL and root parameter $\hat{\theta}_{\text{outer}}$ across three marginal families. The family ranking is stable across marginals (Clayton wins throughout); the parameter magnitudes shift with the marginal (Gumbel least, Clayton most).

	Clayton/Clayton		Gumbel/Gumbel		Frank/Frank	
Marginal	NLL/n	$\hat{\theta}_{\text{outer}}$	NLL/n	$\hat{\theta}_{\text{outer}}$	NLL/n	$\hat{\theta}_{\text{outer}}$
Kaplan–Meier	**7.066**	0.31	7.990	1.12	7.402	1.17
Weibull	**7.720**	0.21	7.976	1.12	8.051	0.93
Lognormal	**7.426**	0.28	8.246	1.12	7.891	1.05

S.2 Retinopathy

Beyond the high-dimensional MIMIC-IV setting, we also evaluate the framework on the diabetic retinopathy paired-eye study, which exercises the small-sample heavy-censoring regime with n=197 and censoring fraction 60.7%, where parameter identifiability is the binding constraint. We split the 197 patients 80/20 at seed 42 to check whether any family overfits in sample, and Appendix S.2.2 then measures how much of the resulting ordering a single split can resolve.

Table 33: Retinopathy: held-out copula log-likelihood for all 10 generator families, with n_{train}=157, n_{test}=40, and seed 42. We fit the Weibull marginals on training data alone; the copula negative log-likelihood $-\ell_{\text{cop}}$ isolates the dependence contribution. The dagger † marks generators with no prior censored-copula implementation.

Family	$-\ell_{\text{train}}$	$-\ell_{\text{test}}$
Nelsen12†	90.422	20.317
Frank	89.423	21.140
InvGauss†	89.503	21.167
Gumbel	89.702	21.203
Nelsen13†	89.492	21.216
Nelsen17†	89.491	21.223
AMH	89.557	21.381
Clayton	89.683	21.420
Joe	90.059	21.490
Nelsen9†	92.827	24.586

For retinopathy, the Nelsen12† generator—the Burr copula—takes the lowest held-out NLL of the 10 families in Table 33, although it converges to its lower parameter bound at $\hat{\theta} \approx 1$ and Appendix S.2.2 shows that its lead does not survive a correction for the nine comparisons. At θ=1 the Nelsen12 generator $\psi(t)=(1+t^{1/\theta})^{-1}$ reduces to Clayton(θ=1), so the Nelsen12 fit is structurally Clayton constrained to $\theta \geq 1$; the unconstrained Clayton MLE in Table 33 sits just below that bound at $\hat{\theta}$=0.61,

and the observed Fisher information gives a full-data Clayton MLE of $\hat{\theta}=0.90\pm0.31$, both consistent with the Nelsen12 boundary fit being Clayton-like. Frank and InverseGaussian† form a tight cluster behind Nelsen12†. The training and held-out orderings agree only at the ends: Frank leads on training NLL, and Nelsen9† trails on both, while Nelsen12† moves from ninth of ten on training to first on test. Nelsen13† and Nelsen17† sit immediately behind the Frank cluster with NLLs that differ in the third decimal, 21.216 against 21.223, at distinct fitted parameters $\hat{\theta}=1.91$ and $\hat{\theta}=2.04$, confirming the two families are not algebraically equivalent. The spread among the top nine families remains modest, with $\Delta-\ell_{\text{test}}\approx 1.2$, consistent with their similarly close training NLLs; Nelsen9† is the sole outlier, drifting toward the independence limit $\hat{\theta}\to 0^{+}$ and fitting poorly.

S.2.1 Standard errors

We compute Hessian-based standard errors from the observed Fisher information returned by `jax.hessian`, with a delta-method projection to the constrained parameter space; the top families show well-identified parameters: Frank $\hat{\theta}=2.25\pm0.62$ at CV 27%, Gumbel $\hat{\theta}=1.25\pm0.08$ at CV 7%, and Clayton $\hat{\theta}=0.90\pm0.31$ at CV 34%. The higher CVs reflect the modest sample size of $n=197$ and heavy censoring of 60.7%; all Hessians remain positive definite, confirming MLEs at proper maxima.

S.2.2 Uncertainty of the retinopathy family ranking

On 40 held-out patients, the dominant variance is not the fit but the partition. The seed-42 split of Table 33 puts Nelsen12† 0.823 nats ahead of Frank and packs the eight families from Frank to Joe into 0.350 nats, all at $n_{\text{test}}=40$, and a bootstrap over that one fixed split sees only which 40 patients land in the test half, never which 157 set the parameters. We therefore refit every family under 5-fold cross-validation with 10 independent repeats, giving 50 folds and 500 censored MLE fits that score every patient under a fit that never saw them, *out of fold*, and we keep the seed-42 split alongside as an exact reproduction of the published protocol. The whole study takes 25.5 minutes on CPU, and every fit converges.

Pre-specified reference. Each interval and each p-value below compares one family against a single reference family, and we choose that reference separately for each of the two analyses that follow, the seed-42 split and the cross-validation, which we call *blocks*. The choice comes from the *training* copula NLL of its own block, never from the held-out scores, and the rule returns Frank in both blocks. Under cross-validation, those training rows are the same 197 patients, because we fit each family on four of the five folds at a time; the held-out scores still play no part in the choice, and we report the simultaneous band alongside as the conservative read. Picking the reference by its held-out score instead reuses those same rows for the choice and for the test, and we measure what the reuse costs at these exact sample sizes. Under an exact null in which all ten families are equally good, the Romano–Wolf step-down rule [Romano and Wolf, 2005] adjusts the nine p-values by resampling the largest *studentised* statistic, an estimate divided by its own standard error, and we report it as the headline decision. With a pre-specified reference, that rule holds its nominal 5% family-wise error—the chance of at least one false rejection anywhere in the table—at 3.3% across all 197 patients and 6.0% on the 40 seed-42 test patients. Once the reference comes from the held-out scores, the same rule drifts to 15.3% and 16.7%, and a single per-comparison interval with no correction at all reaches 69.3% and 68.7%. We report the all-pairs studentised max-t simultaneous band alongside the step-down rule as the conservative read, since the band prices the choice of the best family as well as the nine comparisons.

Interval estimator. The headline cross-validation interval is the Nadeau–Bengio corrected *resampled-t* [Nadeau and Bengio, 2003], a Student-t interval built from the 50 per-fold paired differences, because the obvious alternative understates the variance. Pooling each patient's out-of-fold log-likelihood into one length-197 vector and bootstrapping that vector treats the scores as an independent evaluation sample, but we fit the model on every patient in four of the five folds, so the per-patient differences correlate positively and the independent bootstrap understates the variance of their mean: we measure 91.2% coverage for a nominal 95% interval, with the reported standard error 12–13% too small. The correction scales the per-fold difference variance by the train–test overlap factor $1/J+n_{\text{test}}/n_{\text{train}}$ in place of the naive $1/J$, a $13.5\times$ variance inflation at $J=50$ folds and mean fold sizes $n_{\text{test}}=39.4$ and $n_{\text{train}}=157.6$, so it widens the standard error by $\sqrt{13.5}=3.67$ for every family alike, a factor that J, n_{test}, and n_{train} fix rather than the data. We measure 95.0%

coverage for the corrected interval at five folds and ten repeats, and we keep the pooled bootstrap as a secondary diagnostic. The seed-42 block needs no such correction, because its test rows never touch the fit, so the intervals of Table 6 come from an ordinary paired bootstrap, well calibrated conditionally on that one training split: we resample the 40 test patients $B{=}10{,}000$ times and recompute the mean per-patient log-likelihood difference to the reference, and we measure 93.6% coverage for that procedure at this test size. The cross-validation block is what prices the choice of split itself.

Table 34: Retinopathy under repeated 5-fold cross-validation: 10 repeats, 50 folds, and 500 censored MLE fits scoring all 197 patients out of fold. The left group of columns summarises the 10 repeat-level total copula NLLs; the rank range gives the best and worst rank a family attains over those repeats and p_{best} the fraction of repeats it wins. Those five columns describe this one fixed cohort under different cross-validation partitions, and none of them is sampling variability across cohorts; for sampling variability, read the Nadeau–Bengio interval on the right, whose Δ_{obs} gives the per-patient gap in nats to the pre-specified reference Frank, at four decimal places. Table 6 prices the same comparison on the 40 held-out patients of the seed-42 split alone, as a total; the two protocols score different data and do not agree numerically. No family separates from Frank: every interval covers zero, and every Romano–Wolf step-down p-value exceeds 0.45.

	across the 10 cross-validation repeats					gap to Frank, $\Delta{<}0$ better
Family	$\overline{-\ell}_{\mathrm{out}}\downarrow$	sd	mean rank	rank range	p_{best}	Δ_{obs} [95% CI]
AMH	109.302	0.995	4.2	1–8	0.30	-0.0007 $[-0.0113, 0.0099]$
Nelsen12†	109.357	1.097	3.4	1–8	0.30	-0.0004 $[-0.0148, 0.0140]$
Frank	109.439	1.671	2.7	1–6	0.30	reference
InvGauss†	109.587	1.852	3.6	2–7	0.00	0.0008 $[-0.0016, 0.0031]$
Nelsen13†	109.598	1.642	4.3	2–7	0.00	0.0008 $[-0.0030, 0.0046]$
Nelsen17†	109.609	1.654	4.6	3–7	0.00	0.0009 $[-0.0025, 0.0042]$
Gumbel	110.023	1.904	6.3	1–8	0.10	0.0029 $[-0.0109, 0.0168]$
Clayton	110.047	1.625	7.3	4–9	0.00	0.0031 $[-0.0062, 0.0124]$
Joe	110.735	1.988	8.7	7–10	0.00	0.0066 $[-0.0132, 0.0263]$
Nelsen9†	115.006	0.398	9.9	9–10	0.00	0.0283 $[-0.0201, 0.0767]$

What the two blocks show. On the seed-42 split, exactly one of the nine comparisons against Frank survives the step-down correction, and it is Nelsen9† at the bottom of Table 6, at an adjusted p of 0.022; the Nelsen12† lead does not survive, at $p{=}0.193$ with a 95% interval of $[-1.631, 0.066]$ nats that covers zero. Under cross-validation, none of the nine survives, and the smallest adjusted p is 0.460. The all-pairs band asks a different question, comparing every family against the best-scoring one instead of against Frank and pricing that choice as well as the nine comparisons. Under it, a studentised statistic must exceed 2.968 on the seed-42 block or 2.802 on the cross-validated block, against the 1.96 that a single uncorrected comparison would demand, and no family clears the bar: all ten stay indistinguishable from the best. The two blocks also disagree about which family is best—Nelsen12† on the seed-42 test rows, AMH on mean out-of-fold NLL, and Frank on mean rank—and Table 34 shows the churn behind that disagreement: AMH, Nelsen12†, and Frank each win three of the ten repeats, and at the finer grain of individual folds Nelsen9†, last in Table 6, wins 9 of the 50 folds outright.

Winner's curse. Picking the best of ten families on a test set inflates its apparent advantage even when all ten are equally good, the *winner's curse*. We measure the Nelsen12† advantage over Frank on the seed-42 test rows at 0.021 nats per observation, and the selection offset that ten families of exactly equal quality generate on this cohort's correlation structure under an exact null also comes to about 0.021 nats per observation. The apparent effect is exactly the size of the selection artefact. Two mechanistic details corroborate the reading. The Nelsen12 optimum sits on its parameter boundary at $\hat{\theta}{=}1.000001$, pinned at Kendall's $\tau{=}1/3$, where the generator collapses to Clayton(1), so the family that wins the test rows expresses no shape its neighbours cannot. Nelsen12† also ranks ninth of ten on the seed-42 *training* NLL at 90.422, ahead of Nelsen9† alone, while ranking first on the test rows—the pattern of a family that wins by partition luck rather than by fit.

What the experiment supports. Both blocks reach the same conclusion: one driver fits ten Archimedean generators under censoring, five of which no prior tool implements, and 197 patients do not single out a best family among them.

T Head-to-Head with HACSurv on Framingham

HACSurv [Liu et al., 2025b] is the closest neighbour to our framework in the competing-risks regime (up to five risks plus a censoring node): a nested Gen-AC Archimedean copula with neural density estimator (NDE) marginals, evaluated by an iterative inverse-function-theorem (IFT) solver for the generator inverse. Our framework reproduces HACSurv's per-observation log copula partial to machine precision, isolating their density machinery from ours. We verify the claim in two steps. First, we replicate HACSurv's published Framingham result [Liu et al., 2025b] by running the authors' code as released, with no modification beyond a CUDA-device patch and a SciPy 1.14 `simps`-API shim (Table 5). Second, on the same trained checkpoints, our ACOPULA framework—with the nested Gen-AC generator of Appendix G.1—computes the per-observation copula log partial that enters HACSurv's competing-risks log-likelihood to within machine precision across all five seeds and all three event classes (Table 35); the comparison isolates the density machinery—HACSurv's iterative IFT solve versus our exact compositional Bell polynomial powering—because both paths consume the same generator parameters $M_{\text{outer}}, M_{\text{inner}}, \mu, \beta$ and the same NDE marginal evaluations.

Table 35: Per-observation log copula partial $\log \partial_j C(S_{\text{E1}}, S_{\text{E2}}, S_{\text{C}})$ from HACSurv's iterative IFT solver compared with our ACOPULA framework, on 128 test observations per event class per seed. Here, $S_{\text{E1}}, S_{\text{E2}}$ are the NDE-estimated marginal survivals at the two event times, S_{C} the censoring survival, and $\partial_j C := \partial C/\partial u_j$ the copula partial in variable j (the row's event class). Both paths share the same nested-Archimedean generator and NDE marginals (parameters $M_{\text{outer}}, M_{\text{inner}}, \mu, \beta$ as in Appendix G.1), isolating the density-evaluation step.

	$\max_i \lvert\text{hac}_i - \text{ours}_i\rvert$			$\text{median}_i \lvert\text{hac}_i - \text{ours}_i\rvert$		
Seed	Event 1	Event 2	Censor	Event 1	Event 2	Censor
41	3.8×10^{-12}	2.2×10^{-15}	4.3×10^{-10}	9.3×10^{-16}	8.9×10^{-16}	8.9×10^{-16}
42	2.7×10^{-15}	2.8×10^{-15}	2.2×10^{-15}	$\sim 10^{-15}$	$\sim 10^{-15}$	$\sim 10^{-15}$
43	8.0×10^{-15}	2.4×10^{-15}	2.4×10^{-8}	$\sim 10^{-15}$	$\sim 10^{-15}$	$\sim 10^{-15}$
44	2.1×10^{-15}	2.0×10^{-15}	2.3×10^{-13}	$\sim 10^{-15}$	$\sim 10^{-15}$	$\sim 10^{-15}$
45	3.6×10^{-15}	2.0×10^{-15}	1.8×10^{-15}	$\sim 10^{-15}$	$\sim 10^{-15}$	$\sim 10^{-15}$

The largest seed-wise discrepancy reaches 2.4×10^{-8} on a single seed-43 censored observation whose argument sits near the boundary of the unit cube, where HACSurv's IFT iteration loses digits [Liu et al., 2025b]; across all observations, the median agreement holds at machine precision near 10^{-15}. The replication establishes that our ACOPULA framework reproduces HACSurv's Framingham neural Archimedean copula likelihood. The framework's value-add over HACSurv is the per-variable censoring demonstration at d=53 in Section 5.3, with the neural Archimedean generator extension in Section 5.4—a setting that HACSurv's bivariate-competing-risks formulation cannot express.

End-to-end JAX port with exact Bell polynomial density. Beyond evaluating the copula partial on HACSurv's trained checkpoints, our ACOPULA framework trains the full HACSurv competing-risks model end-to-end without reusing any of their PyTorch code. We port the HACSurv architecture to JAX/Equinox: the shared covariate embedding, three neural density estimator marginals with positive-weight outcome networks for the two risks plus censoring, and the nested Gen-AC Archimedean copula of Appendix G.1. The frailty is $V = \exp(\text{MLP}(U))$ with $U \sim \text{Uniform}(0, 1)$, and the Laplace transform $\psi(t) = \mathbb{E}_U[\exp(-Vt)]$ is estimated at every optimiser step by N_z=100 Monte-Carlo samples of U, matching the simulated-maximum-likelihood (SMLE) objective HACSurv trains. The first-order copula partial routes through ACOPULA's Bell polynomial density path—the same machinery that powers the d=53 MIMIC-IV experiment in Section 5.4—with per-subject censoring masks selecting the single variable differentiated per competing-risks observation. Training reproduces HACSurv's three-stage pipeline: pairwise bivariate copula fits over the three event pairs, inner-generator fitting on samples from the $(1, 2)$ pair, and copula-frozen marginal fine-tuning with early stopping on the validation Conditional-CIF C-index. We report Antolini's time-dependent

C-index and the IPCW-weighted integrated Brier score on the same five HACSurv seeds in the Marginal-SF prediction mode (Table 5).

U S&P 500: Nested Likelihood at d=98

The S&P 500 fit puts an 11-sector hierarchy on 98 equities and measures what the hierarchy costs against a vine copula. We model daily returns over 1,253 trading days (one observation per trading day) as an 11-sector nested copula at $d \in \{50, 75, 98\}$ with 12 parameters, and we convert returns to pseudo-observations by the empirical-CDF rank transform, so the reported likelihoods isolate dependence from marginals. Gumbel attains the best in-sample AIC at d=98 (Table 38); the per-sector θ track within-sector dependence, with Energy and Utilities strongest at θ=1.88 and 1.75 (Appendix V).

Table 36: S&P 500 at d=98: held-out NLL in nats per test day for classical nested Archimedean families and R-vines, as mean and standard deviation over the six chronological folds of Appendix W.2. The column Δ gives the paired gap to the pre-specified reference, the untruncated vine, with its range over the folds.

		$-\bar{\ell}_{\text{test}} \downarrow$	gap to the untruncated vine
Model	$\overline{\lvert\theta\rvert}$	mean $\pm$ sd	Δ [min, max]
Nested Clayton	12	-13.8 ± 4.4	12.51 [9.80, 18.57]
Nested Frank	12	-14.2 ± 3.5	12.15 [7.33, 21.40]
Nested Gumbel	12	-15.2 ± 4.4	11.10 [6.79, 17.94]
R-vine, 10 trees	1,109	-29.5 ± 7.5	-3.20 [-3.83, -2.16]
R-vine, all trees	3,422	-26.3 ± 7.2	reference

A regular vine, or R-vine [Aas et al., 2009, Nagler and Vatter, 2023], factorises the density into conditional pair-copulas, and it wins out of sample by a wider margin than a comparison against the untruncated vine alone would show. Six chronological folds cut the panel between day 753 and day 1,153, fit on the days before each cut, and score the days after: a vine truncated after 10 trees reaches -29.5 nats per test day against nested Gumbel's -15.2, a spread of 14.3 nats. Truncation is what widens the margin, because the untruncated vine overfits: it scores -26.3 on the same days, so the 10-tree vine beats it by 3.20 nats on a third of the parameters. Each remaining row of Table 36 carries its paired gap to that untruncated reference, and nested Gumbel's gap is 11.10 nats, running from 6.79 to 17.94 across the six folds. Seven of the eight gaps keep one sign on every fold, and so do 32 of the 36 pairs among the nine models, so where the cut falls moves the size of a gap far more than its direction (Appendix W.2).

Two properties survive the likelihood gap. First, the nested fit spends 12 parameters against the 10-tree vine's 1,109 and the untruncated vine's 3,422, factors of 92 and 285; because `rvinecopulib` re-selects pair families on each training block, both vine counts average over the six folds. Each of the 12 carries a within-sector or between-sector reading. Second, no vine implementation extends to the per-variable censoring of Section 5.3 (Appendix W). The experiment therefore demonstrates a capability—an exact nested Archimedean likelihood and its exact gradients at d=98, a dimension no prior nested-copula tool reaches—and not a better fit than a vine.

V S&P 500 Sector-Level Tail Dependence

The per-sector θ_{inner} values fitted at d=98 in Appendix U order the sectors by the empirical within-sector co-movement of equity returns, though the upper-tail dependence the Gumbel/Gumbel hierarchy implies overstates the empirical coefficient in every sector. For every pair of stocks within a sector, with U, V the copula-scale pseudo-observations of the two stocks, we compute the empirical upper- and lower-tail dependence coefficients $\hat{\lambda}_U(q) = \hat{\Pr}(U > 1-q \mid V > 1-q)$ and $\hat{\lambda}_L(q) = \hat{\Pr}(U < q \mid V < q)$ at $q \in \{0.05, 0.10\}$, where $\hat{\Pr}$ denotes the empirical relative-frequency estimator over trading days; we then average over pairs within the sector and compare to the Gumbel-implied upper-tail dependence $\lambda_U(\theta) = 2 - 2^{1/\theta}$. Table 37 reports q=0.05; q=0.10 shows the same pattern.

Table 37: S&P 500 d=98 Gumbel/Gumbel fit: per-sector θ_{inner}, theoretical upper-tail dependence implied by Gumbel, and empirical upper/lower-tail dependence at quantile q=0.05, averaged over within-sector stock pairs.

Sector	θ_{inner}	λ_U (Gumbel)	$\hat{\lambda}_U$ (emp.)	$\hat{\lambda}_L$ (emp.)
Information Technology	1.43	0.38	0.28	0.33
Health Care	1.29	0.29	0.20	0.21
Financials	1.50	0.41	0.37	0.38
Consumer Discretionary	1.27	0.27	0.22	0.25
Communication Services	1.24	0.25	0.22	0.26
Industrials	1.28	0.28	0.22	0.24
Consumer Staples	1.37	0.34	0.28	0.28
Energy	1.88	0.55	0.42	0.48
Utilities	1.75	0.51	0.41	0.48
Real Estate	1.45	0.39	0.30	0.33
Materials	1.31	0.30	0.25	0.27

Two qualitative patterns emerge from the table. First, the within-sector θ_{inner} values track empirical co-movement: Energy and Utilities, the two sectors with the strongest physical co-dependence, fit the highest θ_{inner} at 1.88 and 1.75, while Communication Services and Consumer Discretionary—historically diverse mixes of independent business models—fit the lowest. Second, every sector exhibits empirical lower-tail dependence slightly above its empirical upper-tail dependence (average asymmetry $\hat{\lambda}_L - \hat{\lambda}_U \approx 0.03$ at q=0.05), consistent with the well-known crash-driven correlation surge in equity returns. Gumbel imposes $\lambda_L = 0$ by construction and so cannot represent the lower-tail piece exactly; despite that limitation, Gumbel still attains the best in-sample AIC at d=98 among the three classical families (Table 38) and the best held-out likelihood of the three (Appendix W.2), because the upper-tail magnitudes it does capture explain more of the joint structure than Clayton or Frank reach with the same 12 parameters. A practitioner whose data exhibits lower-tail asymmetry should swap Gumbel for Clayton ($\lambda_L > 0$, $\lambda_U = 0$) or for the 180 survival rotation of Gumbel/Joe; the framework treats all options with the same code path.

W Vine Copula Comparison

Vine copulas trade parsimony for flexibility, so a head-to-head against our 12-parameter nested Gumbel quantifies what we give up by collapsing pairwise dependence into a hierarchy. Vine copulas [Aas et al., 2009] model non-exchangeable dependence via pair-copula constructions; Table 38 sets our nested Archimedean model—Gumbel, the best-performing family in Appendix U, alongside Clayton and Frank at d=98—against an R-vine that we fit with `rvinecopulib` [Nagler and Vatter, 2023] on the S&P 500 pseudo-observations, and Appendix W.2 then repeats the comparison on days that no fit has seen.

W.1 In-sample fit

Every model in Table 38 scores the same n=1,253 days its own fit used, the comparison the AIC penalty exists to correct.

The R-vine achieves lower negative log-likelihood at every dimension, reflecting its greater flexibility with $O(d^2)$ pair-copula parameters versus our 12 hierarchical parameters. This flexibility comes at a cost: the R-vine at d=98 spans the 4,753 edges of a full pair-copula construction and fits 3,706 parameters across them, and those parameters lack the direct hierarchical interpretation that nested Archimedean parameters provide as within-sector versus between-sector dependence. The ordering nevertheless survives the held-out evaluation of Appendix W.2, where the best vine leads nested Gumbel by 14.306 nats per test day on days no fit has seen.

Table 38: Nested Archimedean vs. R-vine on S&P 500 data, n=1,253 trading days. $-\bar{\ell}$ is the per-observation negative log-likelihood, lower is better; AIC$/n$ is the per-observation Akaike Information Criterion ($\big(2|\theta| - 2\ell\big)/n$, lower is better). The three classical families appear at d=98, where Gumbel takes the best AIC of the three; Gumbel alone carries the d=50 and d=75 rows. All values in nats.

Model	d	#Params	$-\bar{\ell}$	AIC$/n$
Nested Gumbel	50	12	-9.881	-19.743
R-vine	50	1,191	-20.553	-39.204
Nested Gumbel	75	12	-15.917	-31.815
R-vine	75	2,310	-33.334	-62.982
Nested Clayton	98	12	-20.229	-40.439
Nested Frank	98	12	-20.550	-41.081
Nested Gumbel	98	12	-21.175	-42.330
R-vine	98	3,706	-46.620	-87.325

W.2 Held-out evaluation

The in-sample table rewards whichever model carries more parameters, so we refit the whole comparison on a training block, score it on days no fit has seen, and repeat the exercise over six chronological folds.

Six rolling-origin folds. Each fold cuts the $1{,}253$ trading days at one point, fits every model on the days before the cut, and scores every model on the days after it. Five rolling folds place the origin at days 753, 853, 953, $1{,}053$, and $1{,}153$ and score the 100 days that follow, and a sixth canonical fold trains on the first $1{,}000$ days and scores the last 253. The nested fits and the vines share the fold definitions exactly, so every held-out column compares like with like on identical days. Daily equity returns cluster in volatility, so a random split would surround every test day with training days drawn from the same week and let the fit absorb the local regime; a chronological cut instead asks the question a user of the model faces, namely how well parameters estimated on the past describe a later period. Each cut also leaves its test block one contiguous stretch, so a fold prices a genuine change of regime instead of averaging over interleaved days. The five rolling test blocks tile the last 500 days without overlap, while the canonical block re-scores days that three of the rolling folds also score, so the six folds are neither disjoint nor equally weighted, and we read their spread as a descriptive range over plausible cut points and not as a standard error.

Train-only pseudo-observations. The frozen snapshot `sp500_pseudo_obs_d99.csv` holds full-sample ranks, and slicing that snapshot into a training block and a test block leaks: each training pseudo-observation already counts how many test days fall below it, so the training marginals encode the test period. We therefore rebuild the transform. The training block re-ranks within itself, $u_{ij} = \mathrm{rank}_j(x_{ij})/(n_{\mathrm{train}} + 1)$ over the training rows alone, and each test day passes through the training empirical cumulative distribution function (ECDF), $u_{ij} = \big(\#\{k \in \mathrm{train} : x_{kj} < x_{ij}\} + 0.5\big)/(n_{\mathrm{train}} + 1)$. Re-ranking the snapshot and re-ranking the raw returns give the same answer, because the full-sample rank transform increases strictly within each column, so the frozen snapshot still serves as the input. Six self-checks run on a synthetic fixture at every invocation of the driver, under both the chronological and the random split: the training block equals a direct re-rank of the raw training returns; every value stays strictly inside $(0, 1)$; the transform preserves each column's ordering, strictly on the training block and weakly on the test block; the test block equals a brute-force evaluation of the training ECDF; permuting the test rows leaves the training block untouched; and the corrected training block differs from the naive full-sample slice, which proves the fix changes the answer. Three further assertions guard the S&P panel itself at run time, checking that the mid-rank convention never needs a clip and that both blocks land strictly inside $(0, 1)$. That convention places a test day exactly halfway between the two training ranks that bracket it.

Truncation sweep. Reporting only the untruncated vine understates the baseline, because the untruncated vine overfits. We refit the vine at five truncation levels alongside the untruncated fit on every fold (Table 39): training likelihood improves monotonically with the number of trees kept, while held-out likelihood peaks at ten trees, where the vine beats the untruncated fit by 3.201 nats per

test day on a third of the parameters. No configuration between 11 and 96 trees enters the sweep, so the shape between that peak and the untruncated endpoint stays unmeasured. Truncating at one tree, a Markov tree over the 98 tickers, is the only configuration the untruncated fit beats on every fold.

Table 39: All nine models on the S&P 500 held-out folds at d=98, the full version of Table 36. For each fold, we fit every model on that fold's training block alone, the vines with `rvinecopulib` [Nagler and Vatter, 2023] over its full parametric family set, then score that fold's test block. Every likelihood column carries negative log-likelihood per observation in nats, and the arrow ↓ marks that lower is better; each holds the mean over the six folds and the fold-to-fold standard deviation. The column Δ holds the gap to the untruncated vine, the reference we pre-specify from training likelihood alone; we difference each gap within a fold and then average over the six, so a Δ below zero marks a model that beats the reference. The bracket holds the smallest and the largest of the six fold gaps, a range over cut points and not a confidence interval. The $\approx$ tie mark that Table 4 defines sits on the one row whose range covers zero, the two-tree vine, whose gap changes sign between folds; the other seven gaps keep one sign on all six folds. Because `rvinecopulib` re-selects pair families on each training block, the bar in $\overline{|\theta|}$ marks a mean count over the six folds, here and in Table 36, and the counts run 1,072 to 1,144 for the ten-tree vine and 3,284 to 3,565 for the untruncated one. Table 40 carries the test column fold by fold.

		$-\bar{\ell}\downarrow$, mean ± sd over 6 folds		gap to the untruncated vine, Δ<0 better	
Model	$\overline{\lvert\theta\rvert}$	train	test	Δ [min, max]	tie
Nested Clayton	12	-22.25 ± 0.93	-13.82 ± 4.38	12.510 [9.802, 18.574]	
Nested Frank	12	-22.94 ± 1.15	-14.18 ± 3.54	12.151 [7.330, 21.400]	
Nested Gumbel	12	-23.48 ± 1.09	-15.23 ± 4.44	11.105 [6.791, 17.939]	
R-vine, 1 tree	193	-34.90 ± 1.31	-24.68 ± 6.02	1.653 [0.392, 3.973]	
R-vine, 2 trees	360	-38.57 ± 1.36	-27.57 ± 6.42	−1.239 [−2.438, 0.330]	≈
R-vine, 3 trees	502	-40.05 ± 1.43	-28.65 ± 6.82	−2.323 [−3.444, −1.061]	
R-vine, 5 trees	729	-41.43 ± 1.44	-29.26 ± 7.08	−2.927 [−4.033, −1.851]	
R-vine, 10 trees	1,109	-43.01 ± 1.46	-29.53 ± 7.51	−3.201 [−3.833, −2.164]	
R-vine, all trees	3,422	-50.12 ± 1.97	-26.33 ± 7.16	reference	

Table 40: Held-out $-\bar{\ell}_{\text{test}}$ in nats per test day for each of the nine models on each of the six folds, the raw material behind the mean and the standard deviation of Table 39. Lower is better. The column heading names the fold by its training-block length: the five rolling folds score the 100 days that follow, and the canonical fold scores the last 253. Every fold ranks the three nested fits below every vine, and the ten-tree vine leads on five of the six.

	$-\bar{\ell}_{\text{test}}\downarrow$ by fold, training days					
Model	1,000	753	853	953	1,053	1,153
Nested Clayton	−13.86	−13.12	−14.47	−21.76	−9.39	−10.32
Nested Frank	−13.27	−15.59	−16.70	−18.94	−10.79	−9.79
Nested Gumbel	−14.71	−16.13	−16.87	−22.40	−11.29	−9.96
R-vine, 1 tree	−25.18	−22.53	−23.09	−36.36	−19.60	−21.30
R-vine, 2 trees	−28.05	−25.36	−26.14	−40.01	−22.18	−23.67
R-vine, 3 trees	−29.20	−26.37	−27.68	−41.75	−23.04	−23.89
R-vine, 5 trees	−29.58	−26.96	−28.26	−42.90	−23.17	−24.68
R-vine, 10 trees	−29.70	−26.76	−28.44	−44.10	−23.21	−24.99
R-vine, all trees	−26.44	−22.92	−25.01	−40.34	−20.44	−22.83

Pre-specified reference and fold agreement. Training likelihood picks the reference, and the held-out days play no part in the choice: among the nine models, the untruncated vine leads on training likelihood at -50.12 nats per training day, ahead of every nested fit by at least 26.6 nats, so it serves as the reference for all eight comparisons. Six folds price each gap by how far it moves when the cut point moves, and the bracket in Table 39 reports that movement exactly, as the smallest and the largest of the six fold gaps. Seven of the eight gaps keep one sign on all six folds, so the fold

that scores the quietest period and the fold that scores the loudest one agree about the direction; the two-tree vine is the exception, and its gap to the reference runs from -2.438 to 0.330. Widening the comparison from the reference to every pair leaves the same picture: 32 of the 36 pairs among the nine models keep one sign on every fold, and the four the folds cannot separate are Clayton against Frank, Clayton against Gumbel, the two-tree vine against the untruncated one, and the five-tree vine against the ten-tree one. Six folds support a range and not a coverage statement, because the canonical fold re-scores days that three rolling folds also score and carries a test block two and a half times their length, and a percentile interval needs an exchangeable resampling unit that overlapping folds of unequal length do not supply. Differencing within a fold is what makes the gaps readable at all, and it helps in proportion to how far two models agree about which periods are hard: the fold-to-fold standard deviation of a level runs from 3.5 to 7.5 nats, while the paired gap falls as low as 0.637 nats for the ten-tree vine, a factor of 16.27 against the unpaired combination of the two levels, and the factor drops to 1.56 for nested Frank, whose errors follow the vine's least closely.

Parameter counts. The parameter budget is where the nested model wins on this panel, and the sweep prices that advantage honestly, because the best vine carries a third of the untruncated vine's parameters and the ratio therefore falls from 285 to 92. Nested Gumbel scores -15.23 nats per test day on 12 parameters, 14.306 nats behind the ten-tree vine and 11.105 nats behind the untruncated one, while those two vines carry $1{,}109$ and $3{,}422$ parameters averaged over the folds. A held-out likelihood is the instrument that prices a parameter budget, and the larger model wins it, so the count buys no predictive advantage here. Each of the 12 names one sector or the level that joins them, whereas a vine parameter names a conditional pair, and no arrangement of the $1{,}109$ recovers a within-sector reading. On this panel, the hierarchy therefore buys interpretation and a small parameter budget at a cost of roughly 14 nats per day of predictive likelihood.

W.3 Censoring support and computational pattern

The likelihood gap is only half the comparison, because the censored likelihood that Section 5.3 requires sits outside what vine implementations compute at these dimensions. Standard vine implementations do not natively support censored observations at high dimension. Stöber et al. [2013] extends D-vines to a single shared right-censoring time per observation, sufficient for survival regression with one event but not for the per-variable masks of Section 5.3. Chen et al. [2025] extends C-vines to per-variable right-censoring under informative-censoring assumptions, demonstrated on small dimension; the per-observation cost is a multivariate numerical integration over the censored variables, infeasible at d=53. The computational pattern differs between the two model classes. Vine is density-first—$c(\mathbf{u}) = \prod_e c_e(F_{i_e|D_e}, F_{j_e|D_e})$ is closed-form, but the censored likelihood requires the partial CDF $\partial^{|\delta|}C / \prod_{j:\delta_j=1} \partial u_j$, and recovering this from a density needs multivariate integration over the censored variables, repeated per observation since each mask defines a different integration domain. Nested Archimedean is CDF-first—$C(\mathbf{u}) = \psi(\sum_j \psi^{-1}(u_j))$ is closed-form, the censored variables enter the inner sum as constants, and the partial is an algebraic Bell-polynomial recurrence (Section 3.2). The two model classes therefore serve complementary roles, and the division falls on capability, not on fit: on uncensored panel data at this scale, the vine predicts better, as Appendix W.2 measures, while nested Archimedean copulas reach hierarchical data with the arbitrary per-variable censoring that no vine implementation handles at d=53.

X Hierarchical Kendall Copulas

Hierarchical Kendall copulas (HKCs) sit between vines and nested Archimedean copulas as a third hierarchical construction, but they aggregate clusters through a scalar Kendall transform rather than the full multivariate density—so they cannot supply the analytic gradients or per-variable censoring that our framework provides on the same hierarchical structure. Brechmann [2014] introduced *hierarchical Kendall copulas* as a construction intermediate between vines and nested Archimedean copulas. HKCs aggregate each cluster of variables through the *Kendall distribution function* $K_C(t) = \Pr\{C(\mathbf{U}) \leq t\}$, building the joint distribution from cluster copulas linked by univariate Kendall transforms rather than from generator nesting. Brechmann [2014] derives closed-form HKC density expressions for the special case where every cluster copula is itself Archimedean, drawing on the analytical Kendall function of each generator; for non-Archimedean cluster copulas the density requires multivariate integration, which Brechmann [2014] approximates by sampling rather

than evaluates in closed form. Standard HKC implementations target sampling and pseudo-likelihood estimation through L-BFGS without analytic parameter gradients, and they do not support censored observations. Our ACOPULA framework therefore complements HKCs on the same hierarchical structure with two extra properties: exact analytic gradients through the full multivariate density, and per-variable censoring.

Algorithm 3 Log-domain implementation of Algorithm 1. All floating-point computation uses float64. We write SIGNEDLSE for the signed logsumexp primitive that the subroutine below specifies.

Require: Copula tree at root r with d uncensored leaves, copula-scale observations $\mathbf{u}$ with $u_\ell = F_\ell(x_\ell)$ for raw observation x_ℓ and marginal CDF F_ℓ

Ensure: Joint log-likelihood $\log c(\mathbf{u}) + \sum_{\ell:\,\delta_\ell=1} \log f_\ell(x_\ell)$ (copula log-density plus marginal log-densities for uncensored leaves)

Subroutine NORMALIZE(a): ▷ *Prevent underflow/overflow; keep $a = e^\lambda \hat{a}$*
$s \leftarrow \max_j |a_j|$; $s \leftarrow \max(s, \sqrt{\epsilon_{\min}})$ ▷ $\epsilon_{\min}$ *= smallest positive normal float64 ($\approx 2.2\times10^{-308}$); floor at $\sqrt{\epsilon_{\min}} \approx 1.5\times10^{-154}$ protects $\nabla(a/s)$*
return $\hat{a} \leftarrow a/s$, $\lambda \leftarrow \log s$

Subroutine SIGNEDLSE($\ell_1, \ldots, \ell_n;\ \sigma_1, \ldots, \sigma_n$): ▷ $\ell_j = \log|x_j|$, $\sigma_j = \mathrm{sign}(x_j)$
$m \leftarrow \max_j \ell_j$
$S \leftarrow \sum_j \sigma_j \exp(\ell_j - m)$
return $\log|S| + m$, $\mathrm{sign}(S)$

Step 1. Bell transform ($\alpha^{(c)}$ from child c's $\beta^{(c)}$, d_c uncensored descendants):
$p_0, p_1, \ldots, p_{d_c} \leftarrow \text{JET}\big(\psi_v^{-1} \circ \psi_c,\ t_c,\ d_c\big)$ ▷ $p_j = h_{vc}^{(j)}(t_c)/j!$, *so $p_0 = h_{vc}(t_c) = u_v$ is the primal (unused below)*
Per-index rescaling (removes geometric decay of p_1^j):
$\hat{p}_j \leftarrow \mathrm{sign}(p_j)\ \mathrm{sign}(p_1)^j\ \exp\big(\log|p_j| - j\log|p_1|\big)$ for $j = 1, \ldots, d_c$ ▷ *Writing $\mathcal{P}(z) := \sum_{j=1}^{d_c} p_j\, z^j$, this rescaling produces $\hat{\mathcal{P}}(z) = \mathcal{P}(z/p_1)$*
Polynomial powering (all powers via scan):
$\hat{Q}^{(1)} \leftarrow \hat{P}$
for $k = 2, \ldots, d_c$ **do**
  $\hat{Q}^{(k)} \leftarrow \hat{Q}^{(k-1)} * \hat{P}$ ▷ *Truncated convolution to degree d_c*
Signed logsumexp assembly (incorporates p_1^j factor in log domain):
$w_j \leftarrow \log j! + \log|\beta_j^{(c)}| + j\log|p_1|$, $\sigma_j^{(w)} \leftarrow \mathrm{sign}(\beta_j^{(c)})\ \mathrm{sign}(p_1)^j$ for $j = 1, \ldots, d_c$ ▷ *Per-j log weight; both are k-independent, computed once before the inner loop*
for $k = 1, \ldots, d_c$ **do**
  $\ell_j \leftarrow w_j + \log|\hat{Q}_j^{(k)}|$; $\sigma_j \leftarrow \sigma_j^{(w)} \cdot \mathrm{sign}(\hat{Q}_j^{(k)})$ for $j = 1, \ldots, d_c$ ▷ $\hat{Q}_j^{(k)}$ *= coefficient of z^j in $\hat{\mathcal{P}}(z)^k$*
  $L_k, S_k \leftarrow \text{SIGNEDLSE}(\ell_1, \ldots, \ell_{d_c};\ \sigma_1, \ldots, \sigma_{d_c})$
  $\log|\alpha_k^{(c)}| \leftarrow L_k - \log k!$; $\mathrm{sign}(\alpha_k^{(c)}) \leftarrow S_k$
ref $\leftarrow \max_k \log|\alpha_k^{(c)}|$ ▷ *Reference scale for this child*
$\alpha_k^{(c)} \leftarrow S_k \exp(\log|\alpha_k^{(c)}| - \mathit{ref})$ for $k = 1, \ldots, d_c$; $\alpha_0^{(c)} \leftarrow 0$ ▷ *Internal child: index-0 entry unused in subsequent Cauchy product; differs from censored-leaf base case where $\alpha_0 = 1$ acts as the product identity*

Step 2. Cauchy product ($\beta^{(v)}$ from $\alpha^{(c_1)}, \ldots, \alpha^{(c_M)}$):
$\beta^{(v)} \leftarrow \alpha^{(c_1)}$; $\Lambda \leftarrow 0$ ▷ *Cumulative log-scale*
for $i = 2, \ldots, M$ **do**
  $\beta^{(v)} \leftarrow \beta^{(v)} * \alpha^{(c_i)}$ ▷ *Truncated polynomial product to degree d_v*
  $\beta^{(v)}, \lambda \leftarrow \text{NORMALIZE}(\beta^{(v)})$; $\Lambda \leftarrow \Lambda + \lambda$ ▷ *Re-normalize after each convolution*

Step 3. Root assembly (log-density from root's $\beta^{(r)}$):
$q_0, \ldots, q_d \leftarrow \text{JET}(\psi_r,\ t_r,\ d)$ ▷ $q_k = \psi_r^{(k)}(t_r)/k!$
$\ell_k \leftarrow \log|\beta_k^{(r)}| + \log|q_k| + \log k!$ for $k = 0, \ldots, d$
$\sigma_k \leftarrow \mathrm{sign}(\beta_k^{(r)})\ \mathrm{sign}(q_k)$
logroot, $_ \leftarrow \text{SIGNEDLSE}(\ell_0, \ldots, \ell_d;\ \sigma_0, \ldots, \sigma_d)$

Step 4. Leaf product:
logleaf $\leftarrow \sum_{\ell:\,\delta_\ell=1} \log\big|(\psi^{-1}_{\mathrm{par}(\ell)})'(u_\ell)\big|$ ▷ *Via `jax.grad` of ψ^{-1}; vmap over uncensored leaves*

Step 5. Log-scale accumulation and final result (Steps 1–2 apply at every internal node bottom-up; each ref_c and $\Lambda_{\mathrm{cauchy},v}$ produced below feeds a single global accumulator Λ_{total}):
$\Lambda_{\mathrm{total}} \leftarrow \sum_{v\in\mathrm{internal}}\Big(\sum_{c\in\mathrm{ch}(v)} \mathit{ref}_c\ +\ \Lambda_{\mathrm{cauchy},v}\Big)$ ▷ *Sum every per-child reference scale and every per-node Cauchy normalisation across the whole tree*
return *logroot* $+ \Lambda_{\mathrm{total}} +$ *logleaf* $+ \sum_{\ell:\,\delta_\ell=1} \log f_\ell(x_\ell)$ ▷ *Add marginal log-likelihoods for uncensored leaves*

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract claims (i) Bell-polynomial densities for nested Archimedean copulas via Taylor-mode AD, (ii) per-variable censored likelihood at d=53, and (iii) scaling beyond R's `nacLL` ceiling. Section 3 delivers (i), Section 5.3 delivers (ii) on 85,229 MIMIC-IV admissions, and Sections 5.2 and 5.5 deliver (iii) up to d=98 on real data and d=8,000 in scaling sweeps.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses nesting validity (experiments use one of three valid-by-construction routes—same-family delta, AMH/Clayton shifted-softplus [Hofert, 2010], and the Gen-AC/ACNet Bernstein composition; Appendix G additionally documents a soft-penalty fallback for cross-family or custom generators with no closed-form constraint), identifiability of censored event-time distributions (Tsiatis 1975 obstruction inherited by all copula-survival approaches), the independent-censoring assumption of the MIMIC-IV application, which Appendix S.1 examines, and JIT compile cost. Appendix W explicitly discusses the LL gap to vine copulas as an inherent within-cluster-exchangeability ceiling of nested Archimedean models.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The Bell-polynomial density formula is from Hofert and Pham [2013] and stated in Equation (3) with all required quantities (root, parent, leaves, β recurrence) defined. The McNeil (2008) sufficient nesting condition is stated and the same-family delta parameterisation that satisfies it is in Section 6. The Bell-polynomial recurrence used in Section 3.2 is given in pseudocode (Algorithm 1) and verified empirically against R's `nacLL` to $<10^{-14}$ (Section 5.1 and Appendix J) and Mathematica symbolic derivatives up to $d{=}60$.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All experiments are described with the relevant hyperparameters, optimiser, and data preprocessing in Section 5 and the appendices. The ACOPULA package is public at `https://github.com/thisiscam/acopula`, and the experiment scripts ship as supplementary material with deterministic seeds and frozen pseudo-observation snapshots (e.g., `experiments/data/sp500/sp500_pseudo_obs_d99.csv`) so all numbers in the main text and tables can be regenerated from scratch. The S&P 500 held-out study of Appendix W.2 builds its six rolling-origin folds with `experiments/data/export_sp500_heldout_folds.py`, which rebuilds train-only pseudo-observations from that snapshot on each fold's training block and writes a manifest recording every fold's row ranges, sizes, and checksums beside its fits; `experiments/run_sp500_heldout.py` ships the leakage unit tests behind a `-selfcheck` flag.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may

be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release the ACOPULA package open-source under the Apache-2.0 license at `https://github.com/thisiscam/acopula`; the experiment scripts are attached as supplementary material with a `README` describing exact commands for each experiment in the paper. MIMIC-IV requires credentialed PhysioNet access; we ship the cohort extraction scripts but not the raw data. SP500 pseudo-observations are derived from publicly accessible daily market data, and retinopathy data is publicly distributed with the R `survival` package; both are also bundled with the supplement for one-step reproducibility.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Train/test splits are 80/20 for retinopathy (seed 42, Appendix S.2) and held-out cohort splits for MIMIC-IV (Appendix S.1). Optimiser is L-BFGS-B with exact JAX gradients (Section 5.3). Stage 1 marginals fitted via Kaplan–Meier on training data; stage 2 copula on pseudo-observations. Generator initialisations and parameter bounds are documented in the released scripts.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The parameter recovery study (Table 3) reports empirical SE across $R{=}500$ Monte Carlo replications and verifies it matches the observed-Fisher SE within 2.9%. The HACSurv replication (Table 5) reports mean $\pm$ std across 5 seeds. Retinopathy reports Hessian-based standard errors (Appendix S.2), and Appendix S.2.2 adds repeated 5-fold cross-validation with 10 repeats over all 197 patients, Nadeau–Bengio intervals corrected for train/test overlap [Nadeau and Bengio, 2003], and Romano–Wolf step-down p-values [Romano and Wolf, 2005] across the ten families. For MIMIC-IV, Appendix S.1.2 adds five re-randomised $80/20$ admission-level splits of the fixed $85{,}229$-admission cohort with every model refit from scratch on each, paired percentile intervals over $B{=}10{,}000$ resamples of the $17{,}046$ test admissions within a split, Nadeau–Bengio intervals across splits [Nadeau and Bengio, 2003], and Holm-adjusted p-values [Holm, 1979] alongside an all-pairs studentised simultaneous band; Tables 4, 25 and 30 report the resulting intervals. Appendix W.2 evaluates S&P 500 over six chronological rolling-origin folds with train-only marginals refit on each fold's training block, and reports the fold-to-fold standard deviation of every level together with the smallest and the largest of the six paired fold gaps to a reference pre-specified from training likelihood alone. We read that bracket as a descriptive range over cut points and not as a confidence interval, because the folds overlap and differ in test-block length, so they supply no exchangeable resampling unit; Appendix W.2 states that limitation explicitly.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5 states the recovery, R comparison, SP500, and retinopathy experiments run on a single AMD Threadripper 3960X CPU core; MIMIC-IV runs on an NVIDIA RTX 3090 Ti GPU. Table 27 breaks down per-family compile time and wall time at d=53. Section 6 reports peak memory (1.4 GB at d=98, 700 MB at d=53). The full sweep at $d \in [10, 8000]$ with persistent JAX compile cache fits on a single workstation in under a day.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We use only de-identified, publicly available datasets (MIMIC-IV through credentialed PhysioNet access, retinopathy from Huster et al. [1989], S&P 500 daily returns from public market data). No human subjects research was conducted. No new data collection. The work introduces inference machinery, not predictions deployed against individuals.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper introduces a general-purpose statistical inference framework for nested Archimedean copulas. Its potential downstream uses span survival analysis, financial risk modelling, and reliability engineering — all foundational research with no direct path to deployment-specific harms beyond standard considerations for the application domains themselves. We do not release any predictive model or data resource that introduces new societal-impact considerations.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We release a statistical inference library, not a pre-trained model or scraped dataset. The library has no high-risk-misuse profile beyond standard scientific computing tools.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All baseline software is cited at first use: R `copula` [Hofert et al., 2012, 2013] (GPL), HAC [Okhrin and Ristig, 2014] (GPL), HACSurv [Liu et al., 2025b] (whose published code we used for the head-to-head replication), CopulaCenR [Sun and Ding, 2020] (GPL), `rvinecopulib` [Nagler and Vatter, 2023] (MIT). MIMIC-IV [Johnson et al., 2023] is used under credentialed PhysioNet access. The retinopathy dataset is from Huster et al. [1989], distributed with the R `survival` package. JAX [Bradbury et al., 2018] is Apache-2.0.

    Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset is the ACOPULA Python package. We release it open-source under the Apache-2.0 license at `https://github.com/thisiscam/acopula`; it ships with API docstrings, a `README` quickstart, a documentation site with example notebooks, and worked examples corresponding to the main text experiments (Listing 1).

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or research with human subjects was conducted. We use existing public/credentialed datasets only.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No primary human-subjects research was conducted. MIMIC-IV is de-identified data accessed under PhysioNet's existing data-use agreement; the original IRB process is documented by the dataset's curators [Johnson et al., 2023].

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We used LLM-based coding assistants for paper writing, code editing, and routine implementation scaffolding—usage that falls under the "basic code assistance" and "editing purposes" exemption in the NeurIPS 2026 Main Track Handbook. LLMs are not a methodological component: the Bell-polynomial recurrence is from Hofert and Pham [2013], the Taylor-mode jet application is human-designed, and there is no LLM in the experimental loop, training procedure, or evaluation pipeline. Methodological correctness is independently verified against R's `nacLL` to $<10^{-14}$ (Section 5.1 and Appendix J) and Mathematica symbolic derivatives up to $d{=}60$, so the implementation does not rely on LLM trustworthiness.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.