A Novel Algorithm for Drawing Nested Extreme Value Random Variables*

Wilbur Townsend†

September 2, 2025

This paper presents an algorithm for drawing nested extreme value random variables — *i.e.*, the variable used in the latent variable formulation of the nested logit model. Runtime is linear in both the number of alternatives and the number of nests. An *R* package, `nev`, implements the algorithm.

1 Introduction

The nested logit model is widely used in choice modeling. Given an estimated nested logit model, analytic expressions exist for some objects of interest including demand elasticities, market shares and mean welfare effects of counterfactual policies. However, many other objects have not received an analytic derivation. For example, researchers may be interested in the *distribution* of welfare effects of a counterfactual policy — what proportion of agents are better off, and what proportion are harmed (Townsend & Allan, 2024)? To my knowledge, no formula for that object has been produced.

This paper presents an algorithm for drawing the latent variable from which the nested logit model is derived. With this algorithm in hand, researchers can simulate objects for which no explicit formula has been derived.

Consider a set of alternatives $\mathbf{J}$ which are partitioned into nests $\mathbf{C}$. The *nested extreme value random variable* has CDF

$$\mathbb{P}\left[X \leq \left(x_j\right)_{j\in\mathbf{J}}\right] = \exp\left(\sum_{c\in\mathbf{C}}\left(\sum_{j\in c}\exp\left(\frac{-x_j}{1-\sigma_c}\right)\right)^{1-\sigma_c}\right), \tag{1}$$

where the dependence parameters σ_c determine the correlation between the components of nest c.[1] This distribution is of interest because it is used in the latent variable formulation of the

*Thanks to Jeff Gortmaker and two referees for comments. This research was supported by the Institute for Humane Studies (grant IHS017729).

†University of California, Berkeley. Contact: wilbur.townsend@gmail.com.

[1] Specifically, the correlation between two components in nest c is $2\sigma_c - \sigma_c^2$.

nested logit model: if X has the CDF given by Equation (1) then the probability of choosing any particular alternative k is given by

$$\mathbb{P}\left[k \in \underset{j \in \mathbf{J}}{\arg\max}\left\{\delta_j + X_j\right\}\right] = \frac{\exp\left(\frac{\delta_k}{1-\sigma_{c(k)}}\right)}{D_{c(k)}^{\sigma_{c(k)}} \sum_{c \in \mathbf{N}} D_c^{1-\sigma_c}}; \quad D_c \equiv \sum_{j \in c} \exp\left(\frac{\delta_j}{1-\sigma_c}\right), \tag{2}$$

where the deterministic terms δ_j are sometimes referred to as the 'mean utility' of alternative j and $c(j) \in \mathbf{C}$ is the nest containing alternative j.

In Section 2 I present an algorithm for drawing nested extreme value random variables. In Section 3 I present the *R* package `nev`, in which I implement my algorithm. In Section 4, I validate `nev` by comparing simulated objects to their known values. In Section 5 I conclude by comparing computation time and memory usage to an existing *R* package.

2 The Algorithm

Cardell (1997) shows that nested extreme value random variables can be characterized as

$$X_j = \sigma_{c(j)} \xi_{c(j)} + \left(1 - \sigma_{c(j)}\right) \epsilon_j, \tag{3}$$

where both X_j and ϵ_j have marginal standard-Gumbel distributions. The requisite distribution of each ξ_c is unique but lacks a closed form; by the Convolution Theorem its characteristic function $\phi_{\xi_c}(t)$ is given by

$$\phi_{\xi_c}(t) = \frac{\Gamma\left(1 - \frac{it}{\sigma_c}\right)}{\Gamma\left(1 - \frac{i(1-\sigma_c)t}{\sigma_c}\right)},$$

where $\Gamma\left(1 - \frac{it}{\sigma_c}\right)$ is the characteristic function of $\frac{X_j}{\sigma_c}$, $\Gamma\left(1 - \frac{i(1-\sigma_c)t}{\sigma_c}\right)$ is the characteristic function of $\frac{(1-\sigma_c)\epsilon_j}{\sigma_c}$, and Γ is the gamma function. By the Fourier inversion theorem, the PDF of ξ_c is given by

$$f_{\xi_c}(x) = \frac{1}{2\pi} \int_{\mathbb{R}} e^{-tx} \phi_{\xi_c}(t)\, dt. \tag{4}$$

Our algorithm for drawing nested extreme value random variables is given by:

Algorithm 1:

1. For each nest c, approximate f_{ξ_c} using a numeric approximation to Equation (4).
2. Given our approximations of f_{ξ_c}, draw each ξ_c using inverse transform sampling.
3. Independently draw ϵ_j from a standard Gumbel distribution and so form X_j using Equation (3).

The computation time taken by steps 1 and 2 is linear in the number of nests, while Step 3 is linear in the number of alternatives. It follows that the algorithm as a whole is $O(|\mathbf{C}|)$ and $O(|\mathbf{J}|)$.

3 Implementation in *R*

I implement my algorithm in the R package `nev`, which is available on CRAN. To approximate the Fourier integral (4), I use the `fourierin` package (Basulto-Elias, 2023), which implements the Inverarity (2002) algorithm.

Note that the approximated pdf need not be real-valued, positive, nor close to zero at the boundary of its domain. The `nev` package checks that these approximation failures are not too egregious, and it allows the user to over-ride the default behavior of `fourierin` when they are. Our default parameters ensure that our default tolerance is satisfied for any $\sigma_c \in [0.01, 1]$.[2]

Typical usage of the function *rnev* is as follows:

```
rnev(N, sigma, nests),
```

where the arguments are:

- `N`: the number of vectors;
- `sigma`: the parameters σ_c which determine within-nest correlation, either expressed as a length-$|\mathbf{C}|$ vector, or as a scalar in which case all values of σ_c are assumed to be equal;
- `nests`: a length-$|\mathbf{J}|$ vector of positive integers indicating the nest of each alternative. The number of nests $|\mathbf{C}|$ is assumed to be equal to `max(nests)`.

The output of `rnev` is an `N`-by-$|\mathbf{J}|$ matrix, with each row being a draw from the nested extreme value distribution.

`nev` additionally allows for the keyword arguments `tol`, `lower_int`, `upper_int`, `lower_eval`, `upper_eval`, and `resolution`. `tol` is the tolerance imposed on the requirements that the approximated pdf is real-valued, positive, and near zero at the boundary of its domain. The other keyword arguments are passed directly to the `fourierin` package.

4 Validation

Figure 1 validates `nev` by comparing simulated objects to their known, analytic values. Specifically, I simulate a model containing two nests, each of which contains 4 alternatives. I assume that $\sigma_1 = \sigma_2 = \sigma$. In Panel A I set $\sigma = 0.5$ while in panels B and C I simulate various values of σ. Each specification is simulated using 1000000 draws.

Panel A depicts the marginal CDF of the first component X_1. (As I mentioned above, the marginal distribution of any component is the standard-Gumbel distribution.) Evidently, `nev` is able to closely match the analytic marginal CDF.

[2] In typical applications, the nested extreme value distribution with parameter $\sigma_c \in (0.0, 0.01)$ could be innocuously approximated by treating draws as independent within nests, *i.e.* treating $\sigma_c = 0$. In this case the distribution reduces to the standard-Gumbel distribution.

Panel B depicts the market share of the first alternative $\mathbb{P}\left[1 \in \arg\max_{j \in \mathbf{J}} \{\delta_j + X_j\}\right]$. I set $\delta_1 = 1$ and, for $j > 1$, $\delta_j = 0$. I calculate analytic market shares using Equation (2). Again, `nev` can closely match the correct market shares.

Panel C depicts the correlation between the components X_1 and X_2, which share a nest. This correlation equals $2\sigma_c - \sigma_c^2$. Simulations using `nev` closely match this correlation.

Table 1 compares simulated and analytic values for three different levels of σ_c, otherwise maintaining the same model used to produce Figure 1. Table 1 also assesses how the simulations are affected by different choices of `resolution`, an optional parameter that controls the number of points over which `nev` constructs the numerical PDF. By default, `nev` sets `resolution` equal to 2^{15}. This choice appears to be slightly conservative, with a `resolution` value of 2^{13} yielding similar results.

5 Computation Time and Memory Usage

I now conclude by demonstrating that `nev` can execute quickly, and with low memory usage, even when the number of alternatives is large. For context, I compare the performance of `nev` to the performance of the `evd` package (Stephenson, 2002), which to my knowledge is the only existing *R* package for drawing nested extreme value random variables.[3] I use *R* version 4.0.2, with a 2.60GHz Intel CPU, on a Linux operating system.

Results for various specifications are depicted in Figure 2. Across all specifications I set $\sigma_c = 0.5$ for every nest c, and I draw a single random vector for each specification. In all except the last specification I set the number of nests $|\mathbf{N}| = 2$. (Neither the performance of `nev` nor of `evd` seems much affected by the number of nests, holding fixed the number of alternatives.) I vary the number of alternatives across the x-axes of Figure 2.

In the `nev` package, neither computation time nor memory use are much affected by the number of alternatives, with each draw taking about 0.1 seconds and using about 40 MiB of memory. In contrast, computation time and memory usage for the `evd` package are exponential in the number of alternatives. Thus although `evd` performs more quickly than `nev` when simulating models with few alternatives, `evd` performs more poorly when there are at least 16 alternatives. With 24 alternatives, `evd` takes over 8 minutes and uses over 17 GiB of RAM.

In Table 2 I present the computational performance of additional specifications. Neither the performance of `nev` nor the performance of `evd` is much affected by varying the dependence parameter σ_c. Increasing the number of draws from 1 to 1000 does substantially slow down `evd`. In contrast, most of the time taken by `nev` is used to calculate the numeric Fourier integral;

[3] Note however that `evd` can draw from more general extreme value distributions than just the nested extreme value distribution that I consider.

this integral can be recycled across draws and so increasing the number of draws has a negligible effect on `nev`'s computational performance.

In some applications — for example, to the labor market — the nested logit model is used to model choices from thousands of alternatives (Townsend & Allan, 2024; Azar, Berry, & Marinescu, 2022; Lamadon, Mogstad, & Setzler, 2022). I include in Figure 2 a specification containing a total of 2000 alternatives, evenly spaced across 50 nests. `nev` is able to draw this variable in approximately the same amount of time, and using approximately the same amount of memory, as in the other specifications. For comparison, extrapolating the exponential fit for the `evd` package suggests that using it to draw a nested extreme value random variable for 2000 alternatives would take approximately 10^{587} trillion years.

References

Azar, J., Berry, S., & Marinescu, I. E. (2022). Estimating labor market power. *NBER Working Paper 30365*.

Basulto-Elias, G. (2023). fourierin: Computes numeric fourier integrals [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=fourierin` (R package version 0.2.5)

Cardell, N. S. (1997). Variance components structures for the extreme-value and logistic distributions with application to models of heterogeneity. *Econometric Theory*, *13*(2), 185–213.

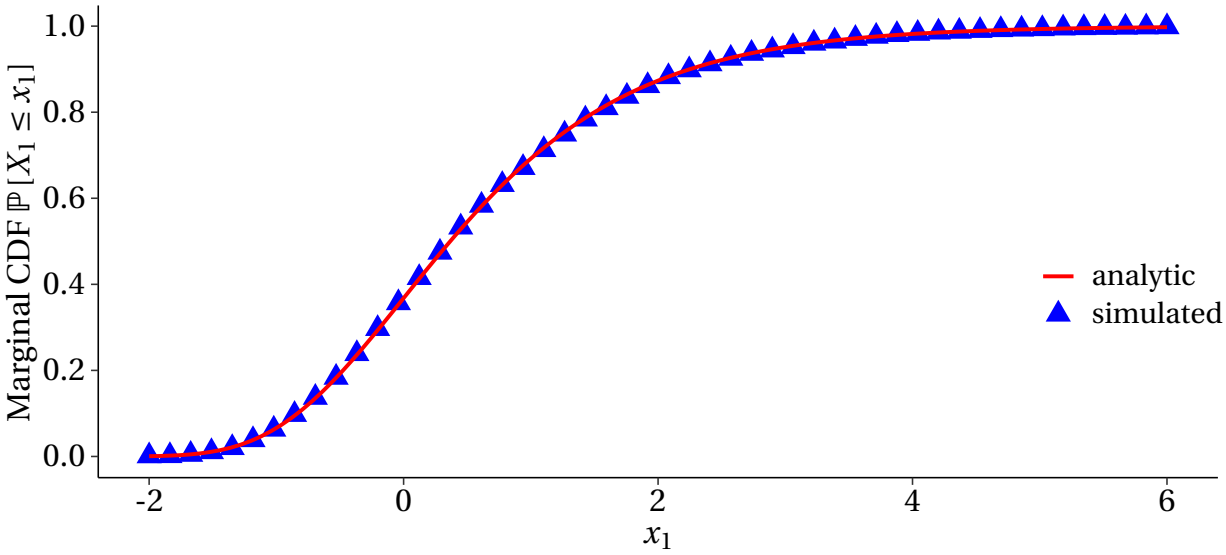
Inverarity, G. W. (2002). Fast computation of multidimensional Fourier integrals. *SIAM Journal on Scientific Computing*, *24*(2), 645–651. Retrieved from `https://doi.org/10.1137/S106482750138647X` doi: 10.1137/S106482750138647X

Lamadon, T., Mogstad, M., & Setzler, B. (2022). Imperfect competition, compensating differentials, and rent sharing in the US labor market. *American Economic Review*, *112*(1), 169–212.

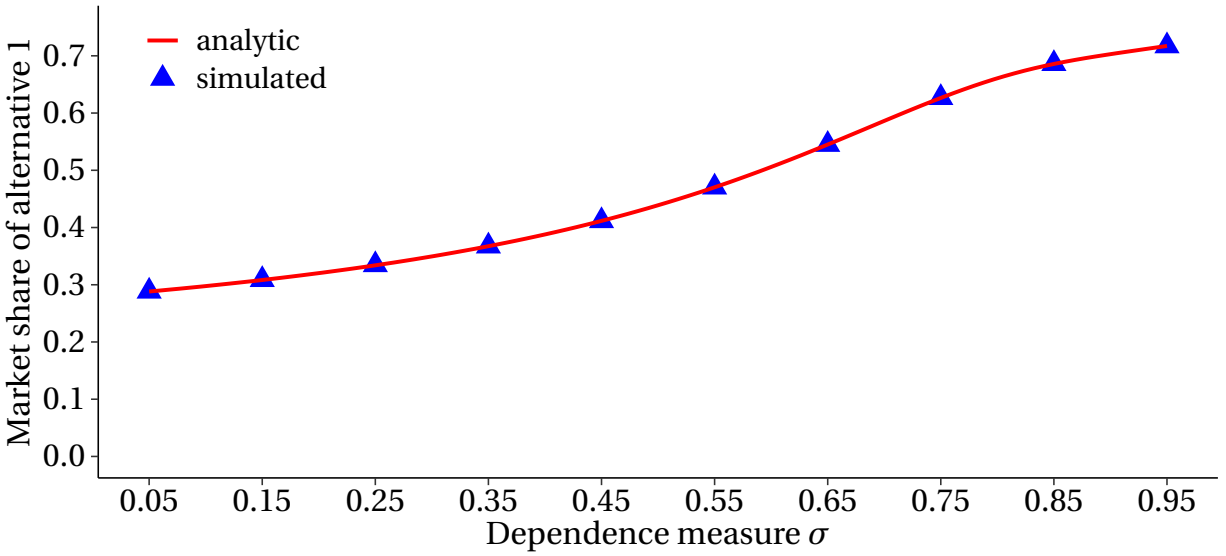
Stephenson, A. G. (2002). evd: Extreme value distributions. *R News*, *2*(2), 31–32.

Townsend, W., & Allan, C. (2024). *How restricting migrants' job options affects both migrants and existing residents.*

Figure 1: Validation

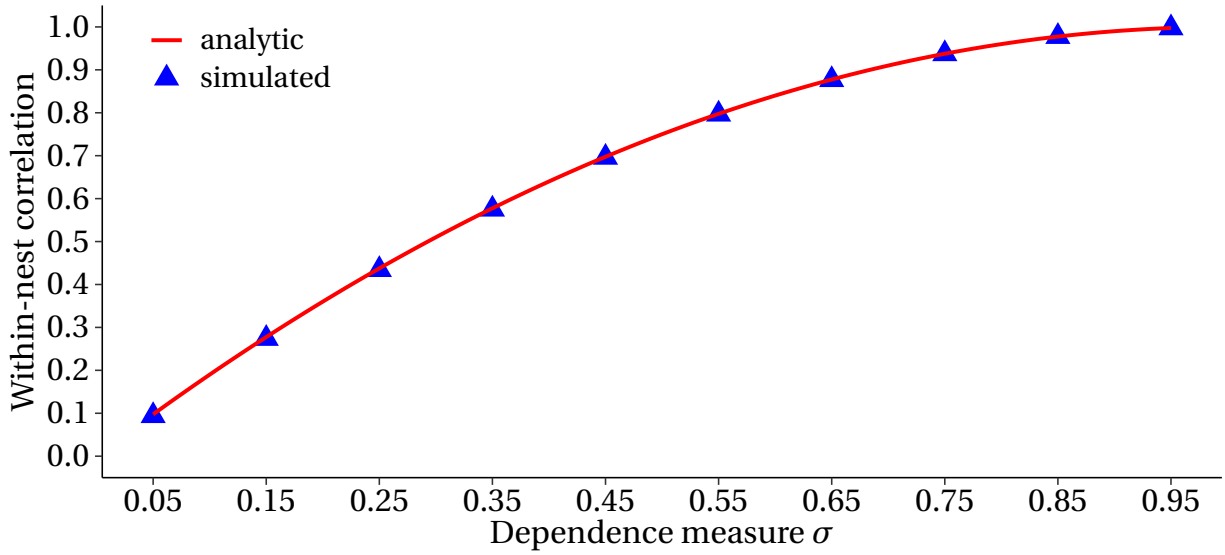


Panel A: the marginal CDF



Panel B: market shares

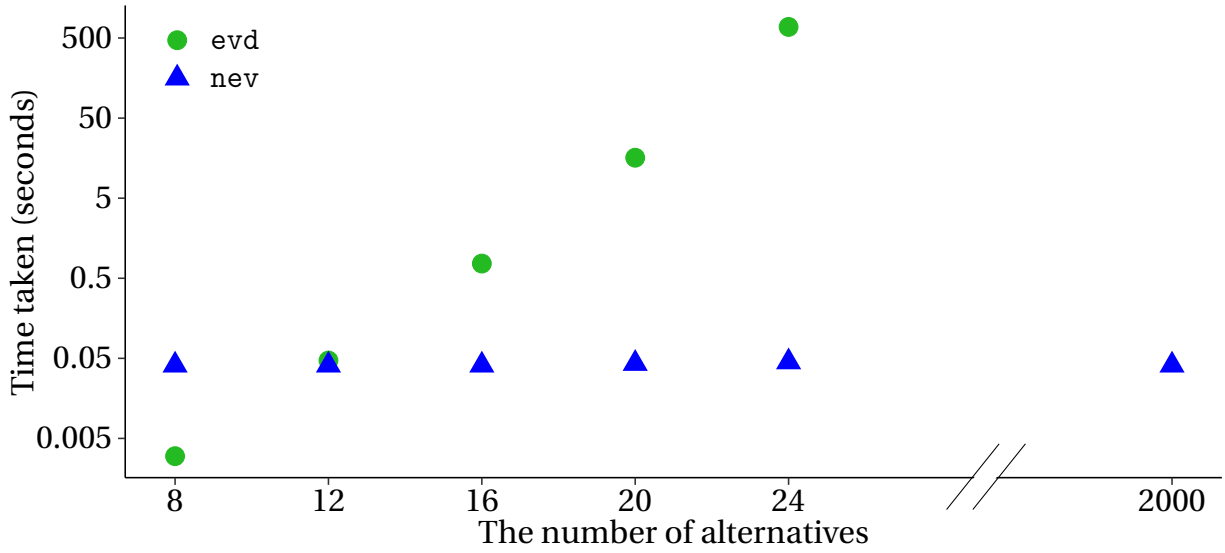
Figure 1 (continued): Validation



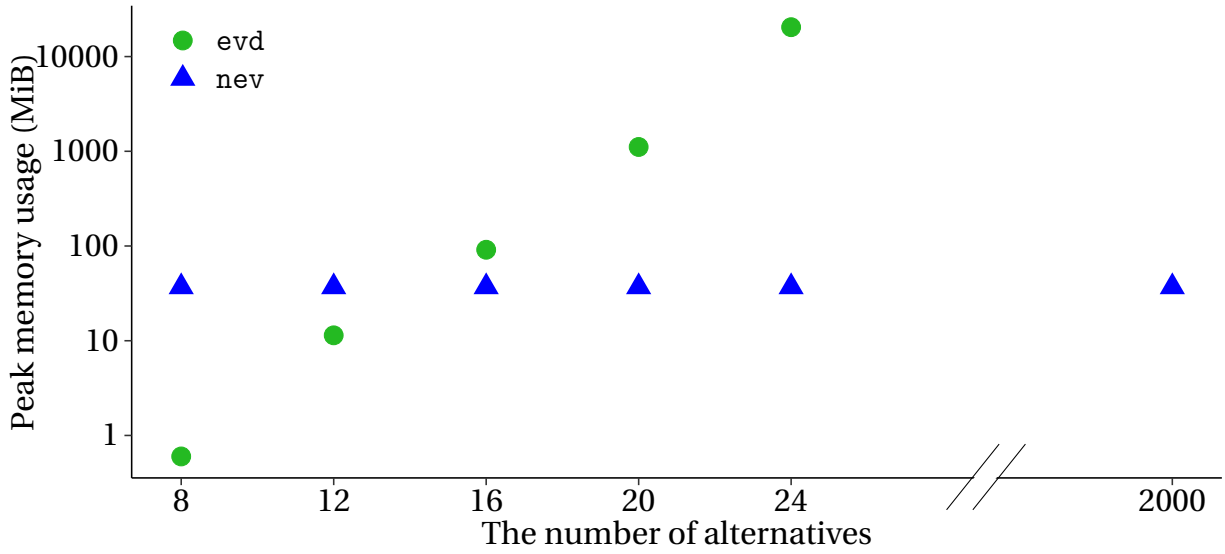
Panel C: within-nest correlation

Figure 1 validates `nev` by comparing the simulated marginal CDF of X_1 (in Panel A), the simulated market share of alternative 1 (in Panel B) and the correlation between X_1 and X_2 (in Panel C) to their true, analytic values. All panels are constructed using a model with 2 nests, each containing 4 alternatives. All panels set $\sigma_1 = \sigma_2 = \sigma$: in Panel A, I set $\sigma = 0.5$, while in panels B and C the value of σ varies across the x-axis. In Panel B I set $\delta_1 = 1$ and, for $j > 1$, $\delta_j = 0$. The analytic marginal CDF is the standard Gumbel distribution, analytic market shares are calculated using Equation (2), and the analytic within-nest correlation equals $2\sigma - \sigma^2$. Simulated values are calculated using 1000000 draws.

Figure 2: Computational Performance



Panel A: Computation Time



Panel B: Memory Usage

Figure 2 presents the computation time (Panel A) and memory usage (Panel B) required to draw a single random vector using the `evd` and `nev` packages. Across all specifications I set $\sigma_c = 0.5$ for every nest c. In all except the last specification I set the number of nests $|\mathbf{N}| = 2$; in the last specification I set the number of nests $|\mathbf{N}| = 50$. I vary the number of alternatives across the x-axes of each panel. The underlying data for Figure 2 is presented in Panel A of Table 2.

Table 1: Validation

	Resolution	Percentiles of the marginal distribution			Within-nest correlation	Market share of Alternative 1
		10th	50th	90th		
Panel A: $\sigma_c = 0.05$						
Truth	—	-0.834	0.367	2.250	0.098	0.288
Simulation	2^5	0.563	4.848	9.022	1.000	0.260
	2^7	0.801	5.222	9.561	0.862	0.259
	2^9	-0.214	4.378	9.034	0.880	0.261
	2^{11}	-0.839	0.357	2.237	0.084	0.288
	2^{13}	-0.835	0.366	2.247	0.096	0.288
	2^{15} (default)	-0.832	0.366	2.252	0.098	0.289
Panel B: $\sigma_c = 0.5$						
Truth	—	-0.834	0.367	2.250	0.750	0.439
Simulation	2^5	-3.527	4.787	12.478	0.990	0.372
	2^7	-3.602	5.112	13.932	0.988	0.378
	2^9	-0.855	0.331	2.150	0.731	0.440
	2^{11}	-0.839	0.359	2.242	0.749	0.439
	2^{13}	-0.835	0.366	2.254	0.751	0.439
	2^{15} (default)	-0.833	0.366	2.252	0.751	0.439
Panel C: $\sigma_c = 0.95$						
Truth	—	-0.834	0.367	2.250	0.997	0.717
Simulation	2^5	-7.561	8.388	23.129	1.000	0.527
	2^7	-8.655	9.225	20.696	1.000	0.557
	2^9	-0.872	0.322	2.181	0.997	0.719
	2^{11}	-0.845	0.355	2.233	0.997	0.717
	2^{13}	-0.835	0.365	2.250	0.998	0.717
	2^{15} (default)	-0.836	0.366	2.248	0.998	0.717

Table 1 validates `nev` by presenting five statistics: the 10th, 50th and 90th percentiles of the marginal distribution of X_1, the correlation between X_1 and X_2, and the market share of alternative 1. Rows titled 'Truth' are calculated using analytic formulae, while other rows are simulated with 1000000 draws. When simulating rows, we compare different values of the 'resolution' parameter that is passed to the numeric Fourier integral package `fourierin`; the bottom row of each panel uses a resolution of 2^{15} which is the `nev` default. In all rows we assume a specification with 2 nests, each containing 4 alternatives, with $\delta_1 = 1$ and, for all $j > 1$, $\delta_j = 0$. In every specification the assumed value of the dependence parameters σ_c are equal between the two nests; panels vary in the value to which this parameter is set.

Table 2: Computational Performance

σ_c	Nests	Alternatives per nest	`evd`		`nev`	
			Time taken (s)	Peak RAM (MiB)	Time taken (s)	Peak RAM (MiB)
Panel A: Drawing a single random vector						
0.05	4	1	0.000	0.1	0.040	37.0
0.05	4	2	0.002	0.6	0.040	37.0
0.05	4	3	0.046	11.4	0.042	37.0
0.05	4	4	0.777	79.5	0.040	37.0
0.05	4	5	15.297	1099.6	0.041	37.0
0.05	4	6	404.512	18472.1	0.046	37.1
0.05	20	50	—	—	0.043	37.0
0.50	4	1	0.000	0.1	0.042	37.0
0.50	4	2	0.003	0.6	0.041	37.0
0.50	4	3	0.047	11.4	0.041	37.0
0.50	4	4	0.763	91.3	0.041	37.0
0.50	4	5	16.001	1110.5	0.043	37.0
0.50	4	6	688.113	20425.6	0.045	37.0
0.50	20	50	—	—	0.041	37.0
0.95	4	1	0.000	0.1	0.040	37.0
0.95	4	2	0.003	0.6	0.040	37.0
0.95	4	3	0.046	11.4	0.042	37.0
0.95	4	4	0.787	92.6	0.041	37.0
0.95	4	5	15.998	1113.9	0.041	37.0
0.95	4	6	898.207	20425.6	0.056	37.0
0.95	20	50	—	—	0.043	37.0
Panel B: Drawing 1000 random vectors						
0.05	4	1	0.001	0.4	0.042	37.5
0.05	4	2	0.019	1.0	0.042	37.7
0.05	4	3	0.347	12.1	0.043	37.8
0.05	4	4	5.904	74.7	0.042	38.0
0.05	4	5	173.393	1304.8	0.047	38.1
0.05	20	50	—	—	0.083	76.9
0.50	4	1	0.001	0.4	0.046	37.5
0.50	4	2	0.022	1.0	0.043	37.7
0.50	4	3	0.368	12.1	0.044	37.8
0.50	4	4	6.242	74.6	0.043	38.0
0.50	4	5	177.512	1301.4	0.043	38.1
0.50	20	50	—	—	0.081	76.9
0.95	4	1	0.002	0.4	0.043	37.5
0.95	4	2	0.020	1.0	0.043	37.7
0.95	4	3	0.348	12.1	0.043	37.8
0.95	4	4	5.880	74.6	0.043	38.0
0.95	4	5	172.754	1331.3	0.044	38.1
0.95	20	50	—	—	0.082	76.9

Table 2 presents the computation time and memory usage required to use the `evd` and `nev` packages. Each row presents times from a different specification, with specifications varying in their dependence parameters σ_c, their number of nests and their number of alternatives per nest. In Panel A I present the time taken and memory required to draw a single random vector, while in Panel B I present the time taken and memory required to draw 1000 random vectors.