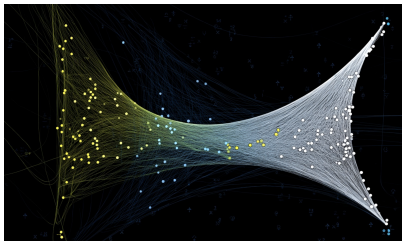
Comonotonicity and Pareto Optimality

Stephen J. Mildenhall

2026-03-11



Mildenhall and Major (2022) discusses kappa functions, likely the most important risk functions you've never heard of. If $X = \sum_i X_i$ is an insurance loss portfolio decomposed by unit i and defined on a probability space (Ω, P), then

$$\kappa_i := \mathsf{P}(X_i \mid X)$$

(i.e., the conditional expectation $\mathsf{E}(X_i \mid X)$, but I'm eschewing E notation in favor of the more transparent and informative P, following advice from Pollard (2002) and practice in many other fields of mathematics). As a random variable,

$$\kappa_i(\omega) = \mathsf{P}(X_i \mid X)(\omega) = \mathsf{P}(X_i \mid X = X(\omega)).$$

M. M. Denuit and Dhaene (2012) call κ_i the Conditional Mean Risk-Sharing (CMRS) allocation.

Kappa functions are important because an insurer is indifferent whether you insure X_i or κ_i. Their total loss is the same in either case. If the insurer prices with a risk functional ρ then $\rho(\kappa_i)$ is often a good premium estimate—it looks stand-alone but gets rid of idiosyncratic and irrelevant process risk. Since κ_i is less risky than X_i, any reasonable ρ should have

$$\rho(\kappa_i) \le \rho(X_i).$$

Less risky is meant in the sense of second order stochastic dominance (SSD), and hence of all three of the non-variance measures in Rothschild and Stiglitz (1970). Spectral risk measures, derived from distortion functions, preserve SSD. This looks like a big pricing simplification, but the simplification is a bit of a cheat since you need to know the multivariate distribution of the X_i to compute κ_i. However, κs are a very helpful mental picture of risk and are often good at explaining how ρ prices.

When ρ is spectral, associated with the distortion g, we write the risk functional as

$$g(X) := \int_0^\infty g(\mathsf{P}(X > t))\, dt$$

rather than $\rho(X)$ to avoid a hidden g or the redundant ρ_g. It is well known there exists a Z, $Z \ge 0$, $\mathsf{P}(Z) = 1$ so that

$$g(X) = \mathsf{P}(XZ), \tag{1}$$

see Mildenhall and Major (2022), terms and conditions apply. Equation 1 leads naturally to the natural allocation of premium

$$g_X(X_i) := \mathsf{P}(X_i Z). \tag{2}$$

The natural allocation can be interpreted as the marginal cost (Delbaen 2000), and it also generalizes the idea of conditional measures like coTVaR. The function Z can be taken to be X-measurable and comonotonic with X, and is morally unique (Cherny and Orlov 2011). Z is called the (linear) contact function for g at X.

If κ_i are all comonotonic with X, then a contact function for g at X is also a contact function for each κ_i. Therefore, since Z is X-measurable,

$$\begin{aligned} g(\kappa_i) &= \mathsf{P}(\kappa_i Z) \\ &= \mathsf{P}(\mathsf{P}(X_i \mid X)Z) \\ &= \mathsf{P}(X_i Z) \\ &= g_X(X_i) \end{aligned}$$

and the stand-alone price of κ_i equals the natural allocation to X_i, a very satisfying result. The comonotonic condition on κ_i holds when all X_i have thin tails (Efron 1965), but not in general. For example, it often fails when one X_i has a thicker tail than the others, see M. M. Denuit et al. (2024) and my blog post discussing it.

In the κ_i allocation of X, each κ_i is less risky than X_i: the allocation is said to dominate the original allocation $X = \sum_i X_i$ in convex order. It is well known that any allocation is convex-order-dominated by a comonotonic allocation. This was first proved in Landsberger and Meilijson (1994) in restricted situations. It would be nice to have a way to convert κ_i into a comonotonic allocation. Alas, none of the existing proofs are explicit enough to allow for an easy algorithmic implementation in practice. Until now: enter the recent paper M. Denuit et al. (2025). It uses alternative approach, based on the theory of majorization and an extension of a result of Lorentz and Shimogaki (1968), to provide an explicit algorithmic construction that is easy to implement. I have added the DDGR algorithm (Denuit, Dhaene, Ghossoub, and Robert) to my risk modeling Python package `aggregate` (in release 0.30).

Example (M. Denuit et al. 2025, sec. 3.2)

Here is how to use `aggregate` to reproduce the example from M. Denuit et al. (2025). The DDGR algorithm is simple but necessarily not quick and can result in a lot of back-tracking for long runs of decreasing κ_i. Therefore, you should install `numba` to pre-compile the numerical implementation. When `numba` is installed, compilation occurs automatically. It provides minutes-to-seconds speedups.

The text below replicates Section 3.2, other than color and figure number changes. Note, the paper uses S where we use X.

Here we provide a simple example that illustrates the algorithmic approach suggested in the proof of Theorem 3.1 [DDGM algo]. Suppose, for the sake of illustration, that $n = 3$. Let $\mathcal{X} \subset L^2_+(\Omega, \mathcal{F}, \mathbb{P})$ denote an ex-ante given collection of potential risks under interest, where $(\Omega, \mathcal{F}, \mathbb{P})$ is a nonatomic probability space. The initial random endowment is the vector $\mathbf{X}_0 = (X_{0,1}, X_{0,2}, X_{0,3}) \in \mathcal{X}^3$, representing the risks faced by the 3 agents individually, before any risk sharing takes place. The pool's aggregate risk is $S := X_{0,1} + X_{0,2} + X_{0,3}$, and the set of feasible allocations is

$$\mathcal{A} = \left\{ \mathbf{X} = (X_1, X_2, X_3) \in \mathcal{X}^3 \;\middle|\; X_1 + X_2 + X_3 = S \right\}.$$

We assume, for the sake of illustration, that the components of $\mathbf{X}_0$ are independent. $X_{0,1}$ and $X_{0,2}$ follow a truncated exponential distribution with parameter $\beta > 0$ on the interval $[0, M]$, for some given $M < +\infty$, with a probability density function f given by $f(x) := \dfrac{\beta e^{-\beta x}}{1 - e^{-\beta M}}$, for $x \in [0, M]$. For some given $N < +\infty$, $X_{0,3}$ follows on the interval $[0, N]$ a truncated mixture of an exponential distribution with parameter $\beta > 0$ and a gamma distribution with parameters $\alpha > 1$ and $\beta > 0$, with equal mixing weights 0.5. The probability density function g of $X_{0,3}$ is then given by

$$g(x) := \frac{(\beta + \beta^\alpha x^{\alpha-1}/\Gamma(\alpha))e^{-\beta x}}{\int_0^N (\beta + \beta^\alpha x^{\alpha-1}/\Gamma(\alpha))e^{-\beta x} dx},$$

for $x \in [0, N]$. For the numerical illustration, we take $\beta = 1/2$, $M = 10$, $\alpha = 8$, and $N = 30$.

Figure 1 provides the probability density function of the aggregate risk S, as well as a plot of the agents' allocations (Figure 2, Figure 3). The initial allocations considered at the beginning of the algorithm are the CMRS allocations $\mathbf{X}_0$ characterized by the functions $s \mapsto \mathbb{E}[X_{0,i}|S = s]$, for agent $i \in \{1, 2, 3\}$ (black lines). We note that the allocation functions of agents 1 and 2 are not increasing in s, and so the CMRS

allocations are not comonotonic, and a fortiori not Pareto optimal. We therefore implement the algorithm presented in the previous section in order to obtain a comonotonic improvement. The final allocation functions (orange lines) become non-decreasing functions in s, thereby leading to a comonotonic and hence Pareto-optimal allocation vector. The algorithm replaced, on an interval including the decreasing parts of agents 1 and 2, the initial functions by constant functions, while preserving the expectations of the allocations and guaranteeing the component-wise convex-order improvement of the allocation vector.

Here is the code to replicate the example. `aggregate` uses parameter $1/\beta$ rather than β for the exponential. Note the use of `splice` in the custom code (DecL) to limit losses from each marginal, and the ease with which the software handles the exponential / gamma mixture for X_3. See aggregate help and the paper Mildenhall (2024) for more about using `aggregate`. The software uses FFTs to convolve the distributions and compute κ_i. Distributions are discretized using $2^{16} = 65536$ buckets and a discretization step $h = 1/512 = 0.001953125$.

```
%config InlineBackend.figure_format = "svg"
import numpy as np
from aggregate import build
from greater_tables import GT
np.seterr(invalid='ignore', divide='ignore')
beta = 2
alpha = 8
M = 10
N = 30
port = build(f'''
port Ex32
    agg X1 1 claim sev {beta} * expon splice [0 {M}] fixed
    agg X2 1 claim sev {beta} * expon splice [0 {M}] fixed
    agg X3 1 claim sev {beta} * gamma [1 {alpha}] wts [0.5 0.5] splice [0 {N}] fixed
''')
GT(port.describe, hrule_widths=(1,0,0), table_font_pt_size=10,
   tikz_scale=0.85)
```

Table 1: Portfolio statistics for Example.

unit	*X*	E[X]	Est E[X]	Err E[X]	CV(X)	Est CV(X)	Err CV(X)	Skew(X)	Est Skew(X)
X1	Freq	1.000			0.00000				
	Sev	1.932	1.932	-41.132n	0.94261	0.94261	136.529n	1.503	1.503
	Agg	1.932	1.932	-41.132n	0.94261	0.94261	136.529n	1.503	1.503
X2	Freq	1.000			0.00000				
	Sev	1.932	1.932	-41.132n	0.94261	0.94261	136.529n	1.503	1.503
	Agg	1.932	1.932	-41.132n	0.94261	0.94261	136.529n	1.503	1.503
X3	Freq	1.000			0.00000				
	Sev	8.842	8.842	-3.444n	0.89216	0.89216	-254.007n	0.545	0.545
	Agg	8.842	8.842	-3.444n	0.89216	0.89216	-254.007n	0.545	0.545
total	Freq	3.000			0.00000				
	Sev	4.235	4.235	-14.906n	1.36784			2.011	
	Agg	12.706	12.706	-14.906n	0.65308	0.65308	-208.549n	0.500	0.500

```
bit = port.density_df.loc[:50].filter(regex='^p_')
bit.plot(figsize=(5,3))
```

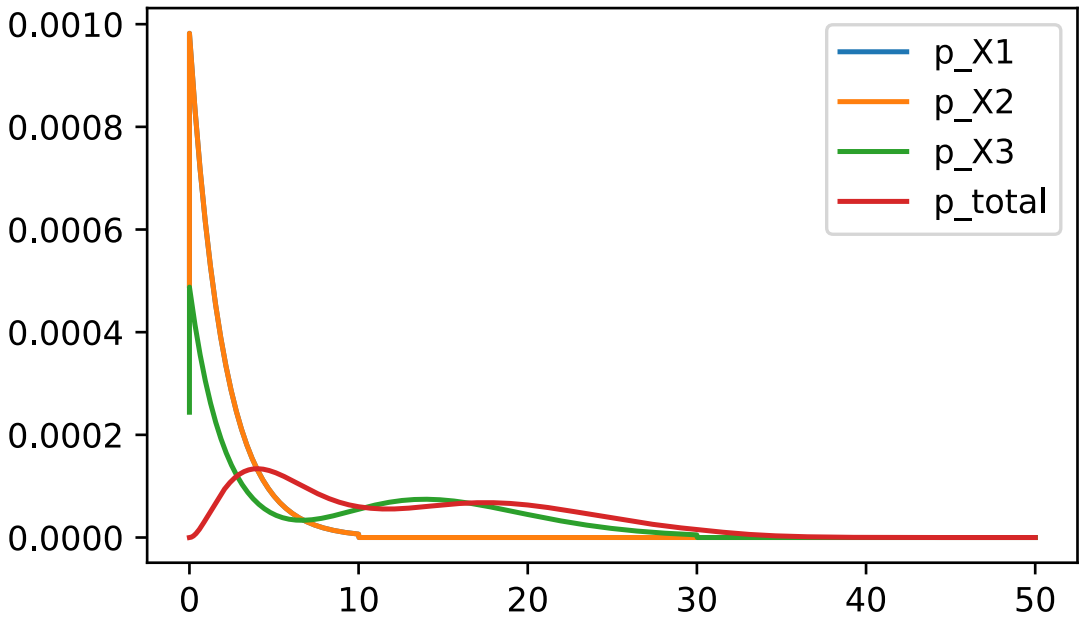


Figure 1: Loss densities by unit.

```
df = port.make_comonotonic_allocations()
df = df.rename(columns=lambda x: x.replace('exeqa_', ''))
df.filter(regex='X1').plot(figsize=(5,3))
```

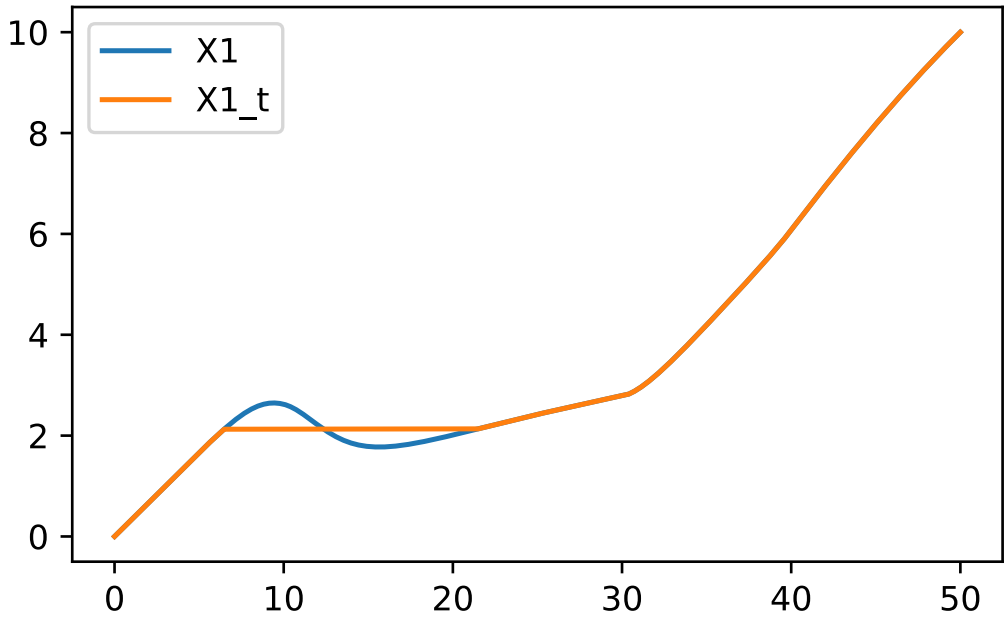


Figure 2: κ_1 and $\tilde{\kappa}_1$.

```
df.filter(regex='X3').plot(figsize=(5,3))
```

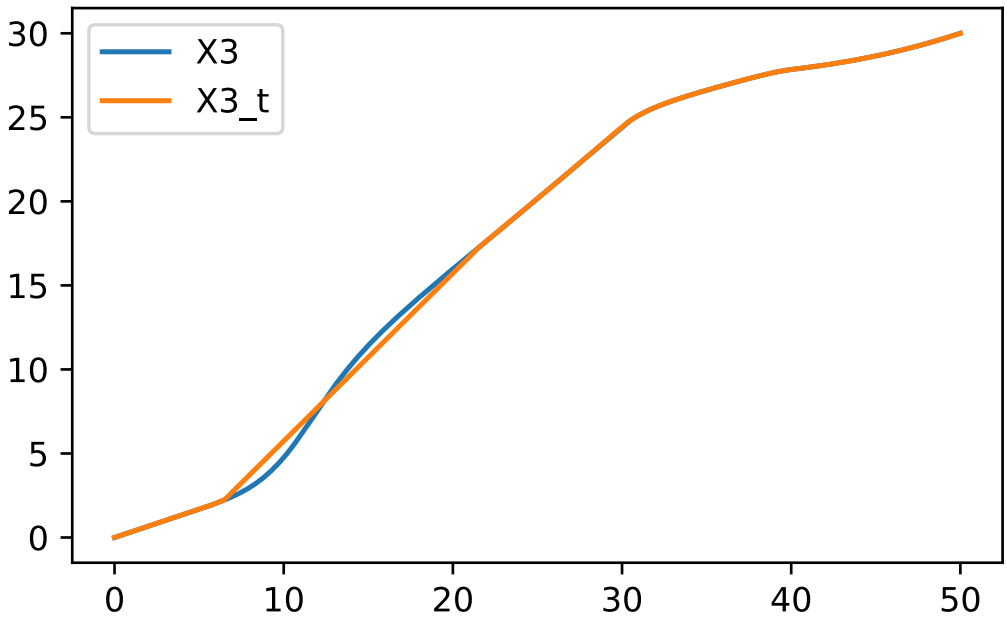


Figure 3: κ_3 and $\tilde{\kappa}_3$.

Source Code

The code for the implementation was produced directly from the paper PDF by Gemini 3.1 in Pro mode. It is included in `aggregate` build 0.30. See the GitHub repo

```
from numba import njit
import numpy as np
@njit
def make_comonotonic_allocations(s_grid: np.ndarray, pdf_s: np.ndarray,
    kappa: np.ndarray) -> np.ndarray:
    """
    Computes a comonotonic convex-order improvement for an allocation matrix.

    Implements the algorithmic convex-order improvement from Theorem 3.1 in
    Denuit et. al.
    Uses a majorization approach based on Lorentz and Shimogaki (1968)
    to flatten monotonicity violations and redistribute mass .

    Reference
    ---------

    Denuit, Michel, et al. "Comonotonicity and Pareto optimality, with application
    to collaborative insurance." Insurance: Mathematics and Economics 120
    (2025): 1-16.

    Parameters
    ----------
    s_grid : np.ndarray
        1D array of length M representing the discretized aggregate sum $S$.
    pdf_s : np.ndarray
        1D array of length M containing the probability mass function of $S$.
    kappa : np.ndarray
        2D array of shape (N, M) where N is the number of individual risks and M
        is the length of s_grid. Represents the initial Conditional Mean
        Risk-Sharing (CMRS) allocations $X_i^0 = \\mathsf{E}[X_i | S]$.
```

```
    Returns
    -------
    np.ndarray
        2D array of shape (N, M) containing the comonotonic allocations
        $\\tilde{f}_i(S)$.
    """
    n, m = kappa.shape
    kappa_tilde = np.copy(kappa)

    # Sweep forward through the aggregate states S
    for k in range(1, m):
        # Calculate local slopes to check for monotonicity
        diffs = kappa_tilde[:, k] - kappa_tilde[:, k-1]

        # Identify components where the allocation decreases as S increases
        violators = np.where(diffs < 0)[0]

        if len(violators) > 0:
            non_violators = np.where(diffs >= 0)[0]

            for i in violators:
                p = k - 1
                mass = pdf_s[k]
                weighted_sum = kappa_tilde[i, k] * mass

                # Scan backward to find the pooling index p that restores
                # monotonicity by creating an integral average (lambda_val)
                # that bounds the previous steps
                while p >= 0 and kappa_tilde[i, p] > (weighted_sum / mass
                    if mass > 0 else kappa_tilde[i, k]):
                    weighted_sum += kappa_tilde[i, p] * pdf_s[p]
                    mass += pdf_s[p]
                    p -= 1

                p += 1

                if mass > 0:
                    lambda_val = weighted_sum / mass
                else:
                    lambda_val = kappa_tilde[i, k]

                # delta represents the mass removed from the violator to flatten it
                delta = kappa_tilde[i, p:k + 1] - lambda_val
                kappa_tilde[i, p:k + 1] = lambda_val

                if len(non_violators) > 0:
                    slopes = diffs[non_violators]
                    sum_slopes = np.sum(slopes)

                    # Compute redistribution weights proportional to positive
                    # slopes to prevent non-violators from breaking monotonicity
                    if sum_slopes > 0:
                        alpha = slopes / sum_slopes
                    else:
                        alpha = np.ones(len(non_violators)) / len(non_violators)

                    # Redistribute the removed mass to the non-violating components
```

```
                    for idx, j in enumerate(non_violators):
                        kappa_tilde[j, p:k + 1] += delta * alpha[idx]

    return kappa_tilde
```

References

Cherny, A. S., & Orlov, D. (2011). On two approaches to coherent risk contribution. *Mathematical Finance*, *21*(3), 557–571. https://doi.org/10.1111/j.1467-9965.2010.00441.x

Delbaen, F. (2000). Coherent risk measures (Pisa Notes). *Pisa Notes*, *24*(4), 733–739. https://doi.org/10.1007/BF02809088

Denuit, M. M., & Dhaene, J. (2012). Convex order and comonotonic conditional mean risk sharing. *Insurance: Mathematics and Economics*, *51*(2), 265–270. https://doi.org/10.1016/j.insmatheco.2012.04.005

Denuit, M. M., Ortega-jimenez, P., & Robert, C. Y. (2024). *Conditional Expectations Given The Sum Of Independent Random Variables With Regularly Varying Densities*. UCLouvain. https://dial.uclouvain.be/pr/boreal/object/boreal/%3A285506/datastream/PDF/_01/view

Denuit, M., Dhaene, J., Ghossoub, M., & Robert, C. Y. (2025). Comonotonicity and Pareto optimality, with application to collaborative insurance. *Insurance: Mathematics and Economics*, *120*, 1–16. https://doi.org/10.1016/j.insmatheco.2024.11.001

Efron, B. (1965). Increasing Properties of Polya Frequency Function. *The Annals of Mathematical Statistics*. https://doi.org/10.1214/aoms/1177700288

Landsberger, M., & Meilijson, I. (1994). Co-monotone allocations, Bickel-Lehmann dispersion and the Arrow-Pratt measure of risk aversion. *Annals of Operations Research*, *52*(2), 97–106. https://doi.org/10.1007/BF02033185

Lorentz, G. G., & Shimogaki, T. (1968). Interpolation theorems for operators in function spaces. *Journal of Functional Analysis*, *2*(1), 31–51. https://doi.org/10.1016/0022-1236(68)90024-4

Mildenhall, S. J. (2024). Aggregate: fast, accurate, and flexible approximation of compound probability distributions. *Annals of Actuarial Science*, 1–40. https://doi.org/10.1017/S1748499524000216

Mildenhall, S. J., & Major, J. A. (2022). *Pricing Insurance Risk: Theory and Practice*. John Wiley & Sons, Inc. https://doi.org/10.1002/9781119756538

Pollard, D. (2002). *A User's Guide to Measure Theoretic Probability*. Cambridge University Press. https://doi.org/10.1017/CBO9780511811555

Rothschild, M., & Stiglitz, J. E. (1970). Increasing risk: I. A definition. *Journal of Economic Theory*, *2*(3), 225–243. https://doi.org/10.1016/0022-0531(70)90038-4