Summary

Stochastic dominance plays a key role in decision-making under uncertainty and quantitative finance. It compares random variables using their distribution functions. This concept helps to evaluate whether one investment, policy, or strategy is better than others in uncertain conditions. By using cumulative distribution functions, stochastic dominance allows decision-makers to make comparisons without assuming specific utility functions. It provides a mathematically rigorous method often used in optimization to maximize returns or minimize risk. Its precision and reliability make it a powerful tool for analyzing complex probabilistic systems.

Despite being a crucial tool in decision-making under uncertainty and quantitative finance, (higher-order) stochastic dominance involves infinitely many constraints, making it *computationally intractable* in practice. Our recent research `@Lakshmanan:2025` addresses this challenge by theoretically reducing the infinite constraints to a finite number. Building on this theoretical foundation, we have developed algorithms to maximize expected returns and minimize risk by satisfying the (higher-order) infinitely many stochastic dominance constraints. The paper aims to establish the mathematical completeness of this reduction and improve accessibility in research domains. However, no concrete, user-friendly implementation of (higher-order) stochastic dominance has been developed. Additionally, the existing prominent theoretical algorithms only discuss stochastic orders *two* and *three*, but not higher orders. Moreover, both, the discussion and implementation of non-integer orders, are absent.

To address this gap, we present *StochasticDominance.jl*, an open-source Julia package tailored for verification and optimization under higher-order stochastic dominance constraints. Designed as a black-box solution, it enables users to input their data and obtain results effortlessly, without requiring extensive technical knowledge.

There is an extensive body of research on the applicability of stochastic dominance across various domains. Numerous scholars, including `@Ogryczak:2002`, `@Dentcheva:2003`, `@Kuosmanen:2004`, `@Kopa:2017`, `@Kopa:2023`, `[@Maggioni:2016; @Maggioni:2019]`, and `@Consigli:2023`, have rigorously explored its significance.
Their work highlights the critical role of stochastic dominance in reducing computational complexity and establishing it as a powerful tool for solving complex optimization problems.
For a comprehensive recent textbook overview, see `@Dentcheva:2024`.

Main features of the package

The StochasticDominance.jl package offers robust functions for verifying higher-order stochastic dominance constraints between two random variables. Additionally, it provides optimization capabilities to construct an optimal portfolio that satisfies and confirms higher-order stochastic dominance constraints. Detailed explanations and usage guidelines are available in the tutorials section.

Technical highlights. StochasticDominance.jl supports two primary objective functions by maximizing expected returns and minimizing higher-order risk measures to achieve the optimal asset allocation while satisfying higher-order stochastic dominance constraints. The package's optimization framework is built around Newton’s method, which efficiently handles the non-linear constraints. To enhance efficiency, we first employ Particle Swarm Optimization (PSO), which approximates the solution over a set number of iterations. In our previous work, @Lakshmanan:2025 initially impose two fixed higher-order stochastic dominance constraints and dynamically introduce additional constraints to ensure dominance. To simplify the process and align with a black-box approach, this package fixes these constraints with additional theoretical backing, eliminating the need for dynamic adjustments. The solutions obtained from PSO serve as the initialization for Newton's method, improving both convergence and accuracy. Below, we provide a concise overview of its key functions.

1. `verify_dominance`: This function checks whether the given benchmark asset, represented as the random variable $X$, and the weighted portfolio asset, represented as the random variable $Y$, exhibit a dominance relationship for the specified stochastic order. This means that $Y$ consistently yields preferable outcomes over $X$ in the specified stochastic order.

2. `optimize_max_return_SD`: This function determines the optimal asset allocation that maximizes expected returns for a given stochastic order (`SDorder`). Additionally, using `optimize_max_return_SD(; plots=true)`, users can generate a pie chart displaying the optimal allocation in percentages, along with the maximized expected returns and benchmark returns. The function also includes the option `optimize_max_return_SD(; verbose=false)`, which allows users to imprint the convergence (or dominance) of the numerical method.

3. `optimize_min_riskreturn_SD`: This function determines the optimal asset allocation by minimizing higher-order risk measures for a given stochastic order (`SDorder`) while also indicating whether dominance is achieved. Additionally, using `optimize_min_riskreturn_SD(; plots=true)`, users can generate a pie chart that visualizes the optimal allocation in percentages, along with the minimizing higher-order risk measure returns. The function also provides the option `optimize_min_riskreturn_SD(; verbose=true)`, allowing users to assess the convergence (or dominance) of the numerical algorithm.

Further implementation details and various examples are available in the package's documentation. The next section presents a typical example of how to use StochasticDominance.jl. for optimization

Example: verification and portfolio optimization

The first example demonstrates how to use the `verify_dominance` function to check stochastic dominance of a specified order. The first step involves loading the necessary library, importing the data, and defining the parameters in Julia as follows:

```
# Load the package
  julia> using StochasticDominance
  # Load the data
  julia> Y = [3, 5, 7, 9, 11] # Random variable Y
  julia> p_Y = [0.15, 0.25, 0.30, 0.20, 0.10] # Probabilities associated
with Y
  julia> X = [2, 4, 6, 8, 10]   # Random variable X
  julia> p_X = [0.10, 0.30, 0.30, 0.20, 0.10] # Probabilities associated
with X
```

Next, define the stochastic order and execute the function as shown below

```
# Define stochastic order
  julia> SDorder = 2;
  # Run the function verify_dominance
  julia> verify_dominance(Y, X, SDorder;p_Y,p_X)
  "Y dominates X in stochastic order 2"
```

This function checks whether $Y$ stochastically dominates $X$ of order `SDorder`. Enabling the `verbose=true` option provides detailed insights into the dominance verification process. In this case `Y dominates X in stochastic order 2`.

Optimization: maximize expected return

Next, we demonstrate how to find the optimal allocation that maximizes the expected return of the portfolios of interest while satisfying stochastic dominance of a given order. This problem falls under non-linear optimization and involves infinitely many constraints. For a more comprehensive technical explanation, we recommend referring to the tutorials.

Now, we load the dataset and define the necessary parameters. We use the dataset from `@Fama:2023`, which is a robust dataset reflecting real-world cases.

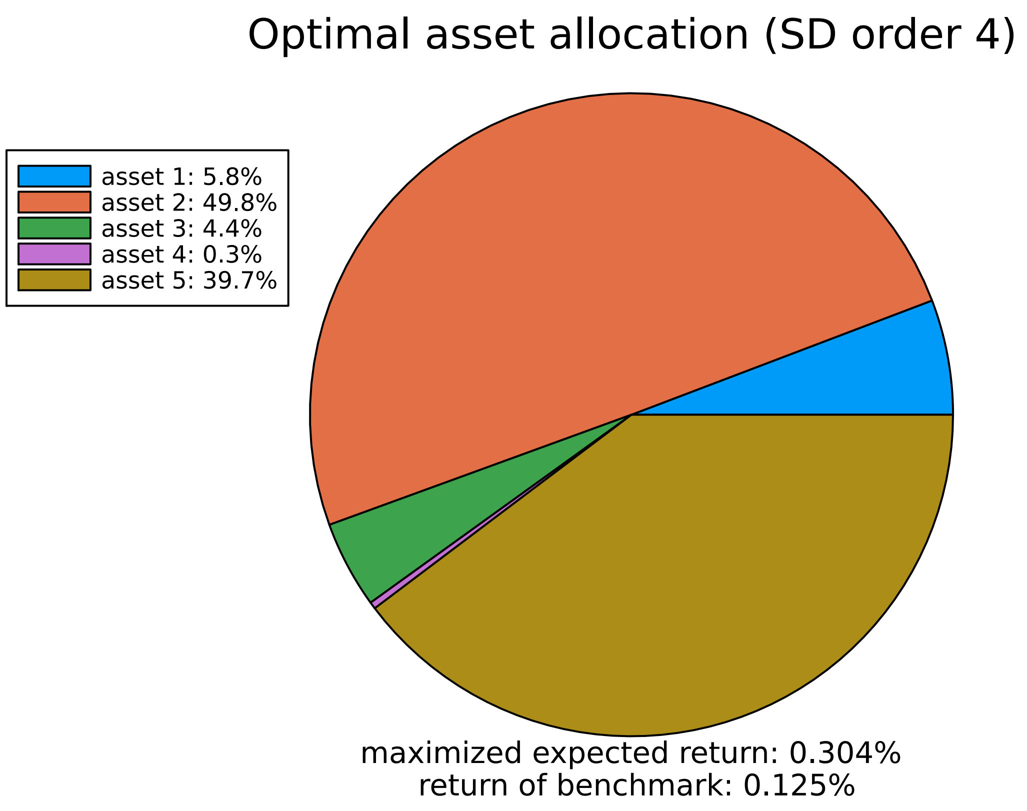
```
# Load the data (Fama and French 2023)
  julia> using Dates, DataFrames
  julia> data = DataFrame(
    Date = Date.([Date("2024-07-01"), Date("2024-07-02"), Date("2024-07-
03"), Date("2024-07-05"), Date("2024-07-08"), Date("2024-07-09"),
Date("2024-07-10"), Date("2024-07-11"), Date("2024-07-12"), Date("2024-07-
15"), Date("2024-07-16"), Date("2024-07-17"), Date("2024-07-18"),
Date("2024-07-19"), Date("2024-07-22"), Date("2024-07-23"), Date("2024-07-
24"), Date("2024-07-25"), Date("2024-07-26"), Date("2024-07-29"),
Date("2024-07-30"), Date("2024-07-31")]),
    Asset_1 = [-1.01, -2.50, -0.38, -1.11, 0.44, 0.05, -0.34, 4.00, 1.76,
1.53, 2.77, 0.71, -2.24, 0.08, 0.35, 2.55, -2.21, 1.67, 0.17, -0.97, -0.11,
0.24],
    Asset_2 = [-0.72, -0.22, 0.27, 0.15, -0.22, -1.49, 0.79, 1.60, 1.04,
0.02, 1.28, 0.69, -1.24, -1.51, 0.29, -0.09, 2.88, -0.21, 1.26, -0.79,
0.79, 1.89],
    Asset_3 = [1.10, -0.86, -0.21, -0.33, -0.64, 1.52, 1.31, 2.53, -0.06,
-3.42, 2.59, -1.44, -1.74, -0.23, -0.54, 0.08, -1.61, 1.05, 2.29, -0.48,
0.98, 0.72],
    Asset_4 = [-1.80, -0.09, 0.41, 2.10, 0.59, -2.59, 2.75, 0.73, 0.41,
-2.50, 0.40, 1.12, -1.72, -2.67, -0.42, -0.39, -2.51, 0.00, 0.93, -0.48,
-1.65, -1.14],
    Asset_5 = [-0.65, 0.64, 0.41, -1.61, 0.33, -0.26, 1.93, 3.72, 2.17,
-0.63, 1.83, 0.94, -1.78, -1.81, -0.25, 1.14, -0.41, 2.21, 2.13, -1.13,
0.07, -1.23])
  julia> xi =  Matrix(select(data, Not(:Date)))'  # Define Portfolio matrix
(d=5 assets and n=22 scenarios)
```

```
  julia> d, n = size(ξ)
  julia> tau = fill(1/d,d)  # equally weights
  julia> xi_0 = vec(tau'*xi) # Define Benchmark
  julia> p_xi = fill(1/n,n)  # Probability vectors for x'*xi, where x is
portfolio weights
  julia> p_xi_0 = fill(1/n,n) # Probability vectors for xi_0
```

Use the following function to compute the optimal allocation (objective: maximize expected return):

```
  # Define stochastic order
  julia> SDorder = 4;
  julia> x_opt, t_opt = optimize_max_return_SD(xi, xi_0,SDorder;p_xi,
p_xi_0,plot=true,verbose=true) # Run the optimization
  "Simplex Constraints residuals: 5.932946379516579e-6
  Stochastic Dominance Constraints residuals: 0.0"
```

From `x_opt`, we obtain the optimal asset allocation. From `t_opt`, we obtain the optimal $t$ ensuring stochastic dominance of the specified order. For further technical details, refer to `@Lakshmanan:2025`. The verbose option provides information about convergence status, simplex constraint residuals, and stochastic dominance constraint residuals for the given order. Enabling `plot=true` generates graphical representations summarizing key insights concisely.



If no portfolio allocation satisfies the stochastic dominance constraint of the given order, the algorithm stops and provides the necessary information.

Optimization: minimizing higher-order risk measures

Next, we demonstrate how to determine the optimal allocation that minimizes higher-order risk measures while ensuring stochastic dominance of a given order for the portfolios of interest. This problem belongs to the class of non-linear optimization. For a detailed definition and technical explanation, we recommend referring to the tutorials.

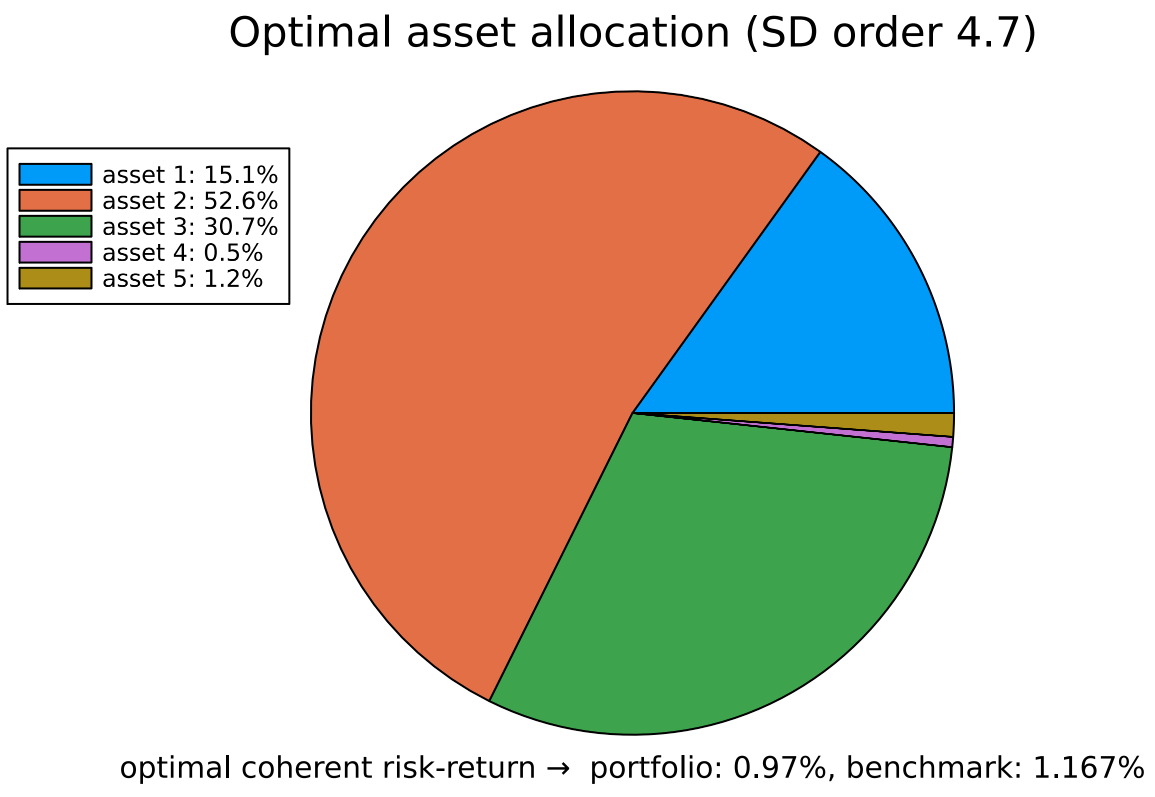
We use the same dataset from the previous section and define the necessary parameters:

```
julia> beta=0.5 #risk parameter
julia> r=2.0 #order of risk measure
```

Use the following function to compute the optimal allocation (objective: minimizing higher-order risk measure):

```
  # Define stochastic order
julia> SDorder = 4.7;
julia> x_opt, q_opt, t_opt = optimize_min_riskreturn_SD(xi,
xi_0,SDorder;,p_xi, p_xi_0,beta,r,plot=true) # Run the optimization
```

From `x_opt`, we obtain the optimal asset allocation. From `q_opt`, we derive the optimal parameter that satisfies the risk measure for the given portfolio. It is important to note that the objective itself is an optimization problem. However, our algorithm is designed to compute both simultaneously in a single execution. Additionally, the algorithm supports non-integer stochastic dominance orders. This feature is not restricted to this single setup but is available across all functions.



References