Extreme Value Theory for Financial Returns with Optimal Tail Selection: A Tutorial for R and Python

Miura Financial Research

Contributors to this tutorial are:

Francesco De Simone
Enrico Schiavina

Outline

Tutorial Overview

In this tutorial, we provide an introductory yet comprehensive overview of Extreme Value Theory (EVT) and its application to modeling financial returns.
We start with a brief review of key concepts in probability and statistics, focusing on the behavior of random variables and properties of probability distributions. Special emphasis is placed on the statistical features of financial returns and the limitations of assuming normality in the tails.
Next, we introduce the core principles of EVT, a powerful framework for modeling extreme events, especially relevant for risk management and finance. We then focus on the Peaks Over Threshold (POT) method and the Generalized Pareto Distribution (GPD), which models exceedances over high thresholds.
A crucial practical issue, threshold selection, is also discussed. We present and implement several automated threshold selection methods from the literature. The code frameworks in this tutorial build upon the R packages `evir` and `tea` for threshold estimation and extreme value modeling.

Random Variables

A **random variable** is formally defined as a measurable function:

$$X : \Omega \to (E, \mathcal{E})$$

where $(E, \mathcal{E})$ is a **measurable space**, for a real valued RV, this space coincide with the real one. This allows for random variables taking values in vector spaces, function spaces, or other structured sets.
The probability space $(\Omega, \mathcal{F}, \mathbb{P})$ consists of:

- Ω: the **sample space**, representing all possible outcomes;
- $\mathcal{F}$: a σ-algebra on Ω, defining the collection of **measurable events**;
- $\mathbb{P}$: a **probability measure**, assigning probabilities to events in $\mathcal{F}$, with $\mathbb{P} : \mathcal{F} \to [0, 1]$, and $\mathbb{P}(\Omega) = 1$.

Random variables are typically categorized by the nature of their range:

$$\text{Discrete: } X \in \{x_1, x_2, \dots\}, \quad \text{Continuous: } X \in \mathbb{R}$$

In financial modeling:

- **Continuous random variables** model quantities like asset returns, interest rates, and price paths;
- **Discrete random variables** model binary or countable outcomes such as defaults, credit ratings, or trade counts.

Probability Distributions

Every random variable is associated with a **probability distribution**. It defines the **law** governing the relationship between the values taken by the random variable and the probabilities assigned to those values.
For discrete random variables, the distribution is described by a **probability mass function (PMF)**:

$$P(X = x_i) = p_i, \quad \sum_i p_i = 1$$

Here, each p_i represents the probability that X takes the value x_i.
For continuous random variables, the distribution is given by a **probability density function (PDF)**:

$$f_X(x) \geq 0, \quad \int_{-\infty}^{\infty} f_X(x)\, dx = 1$$

Probability Distributions

Unlike the PMF, the PDF does not give probabilities directly. Instead, the probability that X falls within an interval $[a, b]$ is given by:

$$P(a \leq X \leq b) = \int_a^b f_X(x)\,dx$$

In fact, for continuous random variables:

$$P(X = x) = 0 \quad \text{for any } x \in \mathbb{R}$$

This is why probabilities are defined over intervals, not at specific points.

Probability Distributions

In both cases, the **cumulative distribution function** (**CDF**) provides the probability that the variable takes a value less than or equal to a given threshold:

$$F_X(x) = \mathbb{P}(X \leq x)$$

In financial applications, the choice of distribution is crucial. Common examples include:

- **Normal distribution**: suitable for simple models assuming symmetry and light tails;
- **Log-normal distribution**: often used to model asset prices, which cannot be negative;
- **Student-t, Variance Gamma, and other heavy-tailed distributions**: employed to account for excess kurtosis, skewness, and extreme events observed in empirical return data.

Expected Value

The most fundamental statistical property of a random variable is its **expected value** (or mean). For a continuous random variable X with density $f_X(x)$, it is defined as:

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x\, f_X(x)\, dx$$

The expected value represents the theoretical long-run average outcome of repeated observations of the variable.

In discrete cases, the expected value is computed as:

$$\mathbb{E}[X] = \sum_i x_i\, P(X = x_i)$$

Variance

The **variance** measures how much a random variable deviates from its expected value. It is the second central moment:

$$\mathrm{Var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$$

A higher variance indicates greater dispersion around the mean. Its square root, the **standard deviation**, is often used for interpretability.

In empirical finance, variance is a fundamental measure of risk.

Skewness

Skewness is the third standardized moment. It measures the asymmetry of a distribution around its mean:

$$\text{Skewness} = \mathbb{E}\left[\left(\frac{X - \mathbb{E}[X]}{\sqrt{\text{Var}(X)}}\right)^3\right]$$

- Positive skew: long right tail (more extreme high values)
- Negative skew: long left tail (more extreme low values)

In finance, skewness is used to detect asymmetry in return distributions, especially downside risk.

Kurtosis

Kurtosis is the fourth standardized moment. It measures the "tailedness" or propensity of a distribution to produce outliers:

$$\text{Kurtosis} = \mathbb{E}\left[\left(\frac{X - \mathbb{E}[X]}{\sqrt{\text{Var}(X)}}\right)^4\right]$$

- High kurtosis: heavy tails, more extreme events than a normal distribution
- Low kurtosis: light tails, fewer outliers

Excess kurtosis is defined as:

$$\text{Excess Kurtosis} = \text{Kurtosis} - 3$$

In finance, high kurtosis reflects tail risk and the possibility of large, unexpected losses or gains.

Modeling Financial Returns

In financial econometrics, asset prices are typically modeled through their returns, which are treated as time-indexed random variables.
Let $\{P_t\}_{t=0}^{T}$ denote the asset price at discrete time points. Returns over one period are defined as:

- **Simple returns:**
$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1$$
- **Log returns:**
$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right)$$

Log returns are often preferred in continuous-time modeling due to their time additivity and analytical tractability.
In both cases, returns are modeled as realizations from a stochastic process $\{R_t\}$, where each R_t is a random variable.

Why Model Returns (Instead of Prices)?

Modeling returns rather than raw prices is a standard practice in financial econometrics for several reasons:

- **Stationarity:** Returns are typically more stationary than price levels, which facilitates statistical inference and forecasting;
- **Comparability:** Returns allow comparison across assets with different price levels and units.

Understanding return dynamics is essential for tasks such as volatility modeling, risk management, and derivative pricing.

Importance of Returns Modeling in Finance

The statistical modeling of asset returns is central to numerous applications in finance:

- **Derivative pricing:**
  - Models like Black-Scholes rely on distributional assumptions for returns.
  - Alternative distributions improve pricing accuracy for options, especially in the presence of skewness or fat tails.
- **Risk measurement:**
  - Measures such as Value-at-Risk (VaR) and Expected Shortfall depend heavily on the assumed return distribution.
  - Misestimating tail behavior can lead to severe underestimation of risk.
- **Portfolio optimization:**
  - Estimating expected returns and covariances is essential for optimal asset allocation.
  - Higher-order moments (skewness, kurtosis) also inform downside risk and investor preferences.
- **Forecasting and trading strategies:**
  - Time-series models for returns help identify market trends and predict volatility.
  - These forecasts drive quantitative trading decisions and hedging strategies.

Accurate return modeling underpins both theoretical developments and practical implementations in quantitative finance.

Why Distributional Assumptions Matter

The choice of the probability distribution for asset returns is crucial in financial modeling, as it affects:

- **Pricing of derivatives**, especially options;
- **Tail risk estimation**, such as Value-at-Risk (VaR) and Expected Shortfall; which determines capital requirements and stress testing;
- **Portfolio optimization**, particularly when **higher moments are included**.

Empirical return distributions often deviate from normality, showing:

- Heavy tails (excess kurtosis);
- Asymmetry (skewness);
- Volatility clustering and time dependence.

Modeling returns with realistic distributions improves the robustness and accuracy of financial decisions.

The Normal Distribution

The **Normal distribution** is a fundamental probability distribution in statistics and finance.

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Its probability density function (PDF) is given by:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Key properties:

- Bell-shaped and symmetric around the mean μ;
- Fully characterized by its mean and variance;
- Skewness = 0 (no asymmetry);
- Kurtosis = 3 (mesokurtic: normal tails);
- **Tails decay exponentially**.

The Normal distribution plays a central role in the classical statistical framework due to the Central Limit Theorem.

Applications in Financial Modeling

In classical finance, asset prices $\{P_t\}$ are often modeled as following **Geometric Brownian Motion** (GBM):

$$dP_t = \mu P_t dt + \sigma P_t dW_t$$

Applying Itô's lemma, the log-price evolves as:

$$\log\left(\frac{P_t}{P_0}\right) \sim \mathcal{N}((\mu - \frac{1}{2}\sigma^2)t, \sigma^2 t)$$

This implies:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right) \sim \mathcal{N}(\mu, \sigma^2)$$

Interpretation:

- Returns over short time intervals are assumed to be normally distributed;
- This forms the basis for many continuous-time pricing models;
- Log-normality of prices implies normality of log-returns.

Applications in Financial Modeling

The Normal distribution is extensively used in financial theory due to its tractability.

Examples:

- **Black-Scholes-Merton Model:**

$$S_t \sim \text{LogNormal} \Rightarrow r_t \sim \mathcal{N}(\mu, \sigma^2)$$

  Assumes log-returns are normally distributed.

- **Value-at-Risk (VaR):**

$$\text{VaR}_\alpha = \mu + \sigma z_\alpha$$

  where z_α is the standard normal quantile.

- **Portfolio theory (Markowitz):**
  - Assumes returns are normally distributed and characterized fully by mean and covariance.
- **Sharpe Ratio:**

$$\text{SR} = \frac{\mathbb{E}[R] - R_f}{\sigma_R}$$

  Based on normality of returns.

These models offer closed-form solutions and intuitive interpretations.

Non-Normal Distributions

In financial data analysis, we often encounter the phenomenon of **non-normality** in asset returns. This characteristic is one of the well-documented *stylized facts* of financial markets, and its presence is routinely confirmed through formal statistical tests, such as the Jarque–Bera or Kolmogorov–Smirnov tests. For an extensive discussion, see Campbell et al. [3], and Brooks [2]. Non-normality in this context is typically associated with the presence of **fat tails**, meaning that extreme events (both positive and negative returns) lead to the presence of large losses or gains, than would be predicted under a normal distribution. To account for this, researchers and practitioners often resort to alternative probability distributions that accommodate heavier tails. Common choices include the *Student's t* distribution, the *Cauchy* distribution, *Generalized Error Distribution*, a mixtures or generalizations of these, as well as: Variance Gamma, Generalized Hyperbolic, and skewed versions of these distributions.
This naturally raises the question: *What do we mean by fat tails, and how can we identify them?* In the following slides, we offer a preliminary exploration of this concept. We will examine both their mathematical forms and their graphical representations to better understand how they differ from the normal distribution, particularly in the behavior of their tails.

Fat Tails via Regular Variation

A rigorous definition of fat tails is based on the concept of **regular variation**. A random variable X is said to have fat tails if its distribution function F satisfies:

$$\lim_{t\to\infty} \frac{1-F(tx)}{1-F(t)} = x^{-1/\xi}, \quad \forall x > 0, \quad \frac{1}{\xi} > 0.$$

This property characterizes distributions with polynomially decaying tails. The heavier the tails, the smaller the value of $\iota = 1/\xi$, and the slower the tail decays.
In this framework, the tails of the distribution converge to a Fréchet form:

$$H(x) = \exp(-x^{-1/\xi}).$$

Such behavior contrasts with exponentially decaying tails (e.g., Gaussian), and it is typical of distributions used in modeling extreme events and financial returns.

Asymptotic Tail Behavior

If X has regularly varying tails, then for large x, the distribution behaves approximately as:

$$\mathbb{P}(X > x) = 1 - F(x) = Ax^{-1/\xi} + o(x^{-1/\xi}),$$

where $A > 0$ is a scaling constant, and $o(x^{-1/\xi})$ represents higher-order terms vanishing faster than the leading term. This expression implies a **Pareto-type** tail:

$$F(x) \approx 1 - Ax^{-1/\xi}, \quad \text{as } x \to \infty.$$

This means the tail behavior of the distribution is governed by the parameter ξ, which quantifies how "fat" the tails are.

Tail Behavior

Therefore, the tail behavior of a distribution is analyzed through the **survival function**, which we recall is:

$$\bar{F}(x) = \mathbb{P}(X > x)$$

Decay Types:

- **Thin tails (e.g., Gaussian)**:

$$\bar{F}(x) \sim e^{-x^2}, \quad \text{exponential or faster decay}$$

- **Fat tails (e.g., power-law)**:

$$\bar{F}(x) \sim x^{-\iota}, \quad \iota > 0, \text{polynomial decay}$$

We can see from the following Figure, that, the slower the decay of the survival function, the fatter are the tails. We can observe this results also by looking at the distribution function F.

Thin vs. Fat Tails

To sum up what we just mentioned in the previous slides, we classify distributions in:

- **Thin-tailed distributions**: tails decay **exponentially** or faster.
- **Fat-tailed (or heavy-tailed) distributions**: tails decay **slower than exponential**.

Moreover, **Nassim Taleb** distinguishes between:

- **Heavy-tailed distributions**: general class with slower-than-exponential decay.
- **Fat-tailed distributions**: extreme subset where tail events *dominate* the distribution's behavior.

Implication: Fat tails are not just about rare events—they define the distribution's core risk structure.

Thin vs. Fat Tails

Thin-tailed distributions (e.g. Gaussian):

- Exponential decay in the tail
- Effectively bounded: extreme values are negligible
- $\iota \to \infty$
- All moments are finite

Fat-tailed distributions (e.g. Power Laws):

- Polynomial decay in the tail
- Large deviations more likely
- $\iota \in (1, 5)$ often
- Some or all moments may diverge:

$$\begin{cases} \iota < 2 \Rightarrow \text{infinite variance} \\ \iota < 1 \Rightarrow \text{infinite mean} \end{cases}$$

Examples of Fat-Tailed Distributions

1. **Lognormal distribution:**
   - Appears power-law for moderate x
   - But decays faster at extreme values
   - Not a true power-law
2. **Student's t-distribution:**
   - Often used to model financial returns
   - For large degrees of freedom: resembles Gaussian
   - For small ν: true fat tails with finite $\iota = \nu$

Both show heavier tails than the Normal, but only the t-distribution approaches power-law behavior asymptotically.

Scale Invariance in Tail Distributions

A distribution is said to be **scale-invariant** in the tails if a multiplicative change in x results in a proportional change in probability:

$$\mathbb{P}(X > x) = L(x) \cdot x^{-\iota}$$

- $\iota > 0$: **tail index**
- $L(x)$: slowly varying function, such that:

$$\lim_{x \to \infty} \frac{L(cx)}{L(x)} = 1, \quad \text{for any } c > 0$$

This implies a **linear slope in the log-log plot** of the survival function — a hallmark of power laws.

Scale-Invariance vs Scale-Variance

Scale-Invariant (Power Laws):

- Constant rate of decay on log-log scale
- Probability of extreme events decays slowly
- Examples: Pareto, Zipf, Cauchy, some Student-t

Scale-Variant:

- Decay accelerates with x
- Tails curve downward in log-log scale
- Examples: Gaussian, Lognormal

Impact of Tail Index ι

Lower tail indices produce fatter tails and more extreme outcomes:

$$\mathbb{P}(X > x) \sim x^{-\iota}$$

- $\iota < 2 \Rightarrow$ Variance is infinite
- $\iota < 1 \Rightarrow$ Mean is infinite

This does *not* mean tail events are frequent — but when they occur, they are orders of magnitude larger.

Pareto Example

$$\mathbb{P}(X > x) = \left(\frac{x_{\min}}{x}\right)^{-\iota}, \quad x \geq x_{\min}$$

Modeling Tails

Understanding and modeling the behavior of distribution tails is a fundamental challenge in financial risk management. It is precisely in the tails where the most extreme—and often the most damaging—events occur: sharp market crashes, liquidity crises, extreme credit defaults, and other rare but impactful scenarios. These events are by definition infrequent, but when they happen, they can lead to disproportionately large losses.
For this reason, relying solely on models that focus on the center of the distribution or assume normality can be dangerously misleading. Standard risk metrics, like Value at Risk (VaR) based on Gaussian assumptions, tend to significantly underestimate the likelihood and magnitude of extreme losses. Thus, there is a clear need for approaches that focus explicitly on modeling the tails of financial distributions.
A natural idea might be to assume a specific distributional form for the tails and estimate the probability of extreme losses from there. However, this is easier said than done. In many financial applications, the underlying distributions exhibit heavy tails, often with tail indices (ι) smaller than 2. In such cases, the variance—and sometimes even the mean—of the distribution may not exist. **This invalidates many classical statistical techniques, which rely heavily on the existence of finite moments**, for instance the modern portfolio theory.

Modeling Tails

This is where Extreme Value Theory (EVT) comes into play. EVT provides a rigorous mathematical framework to model and quantify extreme events without making strong assumptions about the overall distribution. By focusing only on the behavior of the extremes (or exceedances above a threshold), EVT allows us to make probabilistic statements about rare events in a statistically sound way.
In the context of financial risk, EVT has become an essential tool for estimating tail-related quantities such as extreme quantiles, tail-based Value at Risk, and Conditional VaR (Expected Shortfall). While it does not eliminate the uncertainty inherent in modeling extremes, it provides one of the most robust and theoretically justified approaches to dealing with the tails—where traditional models often fail.

Applications of EVT in Finance

- **Tail Risk Modeling**
  EVT allows for accurate estimation of the probability and impact of rare, extreme losses in financial markets.
- **Value at Risk (VaR) and Expected Shortfall (ES)**
  EVT provides tail-sensitive estimates of VaR and ES, especially under heavy-tailed return distributions.
- **Stress Testing**
  Used to simulate extreme market conditions beyond historical observations.
- **Portfolio Risk Management**
  Helps identify assets or strategies particularly exposed to extreme events.
- **Credit and Operational Risk**
  EVT is used to model extreme losses due to defaults, fraud, or operational failures.
- **Insurance and Reinsurance Pricing**
  Tail modeling is crucial for pricing catastrophic financial events and designing risk transfer mechanisms.

EVT and Tail Behavior

Extreme Value Theory (EVT) focuses on modeling the distribution of **extreme outcomes** — the most severe losses or gains — rather than the full distribution. This is especially important for measuring **tail risk** in financial data.
EVT assumes that the tail of the return (or P&L) distribution converges asymptotically to one of three canonical forms:

1. **Weibull**: – thin tails with a finite upper bound (bounded support)
2. **Gumbel**: – medium tails, exponential decay (e.g., normal, log-normal)
3. **Fréchet**: – heavy tails, power-law decay; common in finance

These types are determined by the **shape parameter** ξ in the Generalized Extreme Value (GEV) or Generalized Pareto (GPD) distributions.

EVT and Tail Behavior

In the heavy-tailed (Fréchet) case, the shape parameter $\xi > 0$ is especially important, as it indicates the presence of power-law decay in the tails.

- Financial return data often exhibit Fréchet-type behavior, where EVT provides more realistic and robust estimates of tail risk than traditional models that assume finite variance (e.g., Gaussian).
- Understanding ξ and ι helps in selecting appropriate models for risk management, stress testing, and capital allocation under extreme market scenarios.

The Generalized Extreme Value Distribution (GEV)

The theorems by Fisher and Tippett (1928) and Gnedenko (1943) [8] are fundamental results in Extreme Value Theory (EVT). They describe the behavior of the maximum of a sample of properly normalized independent and identically distributed (i.i.d.) random variables.
Specifically, as the sample size T grows to infinity, the distribution of the normalized maximum converges to one of three possible limit distributions: Weibull, Gumbel, or Fréchet.

An alternative but equivalent way to express these results is through the concept of the **Maximum Domain of Attraction** (MDA).
The MDA of a distribution refers to the set of all possible limit distributions for the normalized maxima of samples drawn from it, as the sample size tends to infinity. Understanding the MDA helps in classifying the tail behavior of the underlying data and selecting appropriate models for extreme value analysis.

The Generalized Extreme Value Distribution (GEV)

Let $\{X_1, X_2, \ldots, X_T\}$ be i.i.d. random variables, and define the maximum as $M_T = \max(X_1, \ldots, X_T)$. The standardised distribution of maxima, M_T, is

$$\lim_{T \to \infty} \Pr\left\{ \frac{M_T - a_T}{b_T} \leq x \right\} = H_\xi(x)$$

where $a_T = T\,\mathbb{E}[X_1]$ and $b_T = \sqrt{\mathrm{Var}(X_1)}$ are normalizing constants chosen appropriately.
Then the limiting distribution, $H(\cdot)$, of the maxima as the **generalised extreme value (GEV) distribution** is:

$$H_\xi(x) = \begin{cases} \exp\left\{-(1+\xi x)^{-1/\xi}\right\}, & \xi \neq 0 \\ \exp\left\{-e^{-x}\right\}, & \xi = 0 \end{cases}$$

where the shape parameter ξ determines the type of tail behavior:

- $\xi > 0$: **Fréchet** distribution (heavy-tailed),
- $\xi = 0$: **Gumbel** distribution (light-tailed, exponential decay),
- $\xi < 0$: **Weibull** distribution (bounded tails).

Generalized Extreme Value Distribution (GEV)

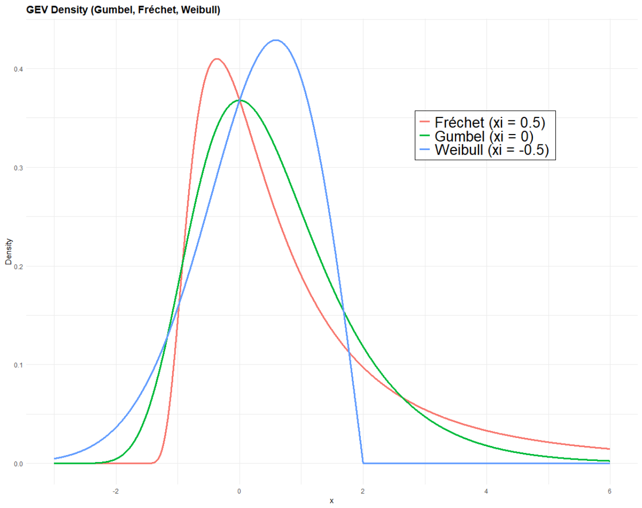

Figure: GEV distribution: Fréchet, Gumbel and Weibull case.

EVT Methods

If we want to implement EVT, two possible approaches can be considered:

- Block Maxima;
- Peak Over Threshold.

Block Maxima
This method is grounded in the theory of regular variation, where the Generalized Extreme Value (GEV) distribution is estimated by dividing the data into non-overlapping blocks and considering only the maximum observation within each block for inference.

Although conceptually straightforward, this approach is data-inefficient, as it discards all non-maximal values. Consequently, a large sample size is typically required to obtain stable and accurate parameter estimates.

EVT Methods

Peak Over Threshold
This approach is generally preferred and serves as the foundation of the methodology presented below. It relies on modeling all large observations that exceed a high threshold, thereby allowing for more efficient use of data on extreme values.
There are two main methodologies within the Peaks Over Threshold (POT) framework.

- Fully parametric: which typically involves using the Generalised Pareto Distribution (GPD);
- Semi-parametric: based on Hill estimator.

Peak Over Threshold

Let X be a continuous random variable, and choose a high threshold u.

We are interested in the behavior of large values of X, specifically the excess over the threshold:

$$Y = X - u \quad \text{given that } X > u.$$

The conditional distribution of exceedances is:

$$F_u(x) = \Pr(X - u \leq x \mid X > u), \quad x > 0.$$

When the threshold u is set at the Value-at-Risk (VaR) level:

- $F_u(x)$ represents the distribution of losses that exceed VaR.
- This is also referred to as the **conditional excess loss** or **expected shortfall distribution**.

Understanding $F_u(x)$ is crucial for modeling the tail of the loss distribution.

Generalized Pareto Distribution

Under general regularity conditions, the distribution of exceedances over a threshold u converges to a Generalized Pareto Distribution (GPD) as $u \to \infty$:

$$F_u(x) \longrightarrow G_{\xi,\beta}(x)$$

- The GPD is defined as:

$$G_{\xi,\beta}(x) = \begin{cases} 1 - \left(1 + \dfrac{\xi x}{\beta}\right)^{-1/\xi}, & \text{if } \xi \neq 0 \\ 1 - \exp\left(-\dfrac{x}{\beta}\right), & \text{if } \xi = 0 \end{cases}$$

This result forms the basis of the POT approach for modeling tail behavior. Key references: Balkema and de Haan (1974), Pickands (1975), Davison and Smith (1990).

Generalized Pareto Distribution

The PBdH [1], [12] theorem formalizes the convergence to the GPD:

$$\lim_{u \to x_F} \sup_{0 \le x < x_{F-u}} |F_u(x) - G_{\xi,\beta}(x)| = 0$$

- That is, for any distribution F in the maximum domain of attraction of an extreme value distribution G_ξ, the excess distribution $F_u(x)$ over a high threshold u is well-approximated by a GPD.
- This provides the theoretical justification for using the GPD in modeling threshold exceedances.

Some aspects to consider:

- The GPD models the tail behavior of a wide range of distributions.
- The exceedances over a high threshold follow a homogeneous Poisson process.
- The maximum of a Poisson number of i.i.d. exceedances converges to a Generalized Extreme Value (GEV) distribution.

Generalized Pareto Distribution

The Generalized Extreme Value (GEV) distribution arises as the limiting distribution of properly normalized sample maxima. In contrast, the Generalized Pareto Distribution (GPD) emerges as the limiting distribution for exceedances above a sufficiently high threshold.

Importantly, both the GEV and the GPD share the same **tail index**, which governs the heaviness of the tail.
Moreover, the parameters of the GEV distribution can be inferred indirectly by maximizing the log-likelihood function of the GPD, leveraging the connection between threshold exceedances and block maxima.

- ξ controls the tail heaviness (**shape** parameter)
- $\beta > 0$ is the **scale** parameter

The domain of x depends on ξ:

$$x \in [0, \infty) \quad \text{if } \xi \geq 0; \qquad x \in \left[0, -\frac{\beta}{\xi}\right] \quad \text{if } \xi < 0$$

Generalized Pareto Distribution

Both parameters must be estimated from data when fitting the GPD. Depending on the value of the shape parameter ξ, the GPD corresponds to well-known tail behaviors:

- $\xi > 0$: **Fréchet-type** tails (heavy-tailed)
- $\xi = 0$: **Exponential** tails (Gumbel limit)
- $\xi < 0$: **Bounded** tails (Weibull-type)

These correspond to the three domains of attraction in extreme value theory.

Generalized Pareto Distribution

We can observe the GPD density for different shape ξ parameters.

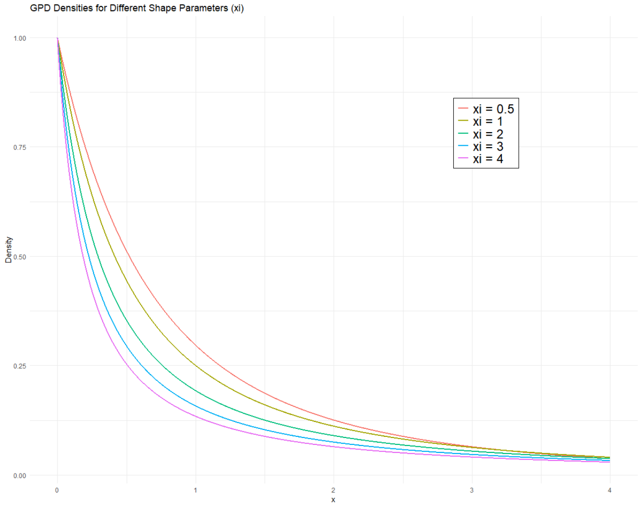

Figure: GPD Density for different shape parameters: $\xi = 0.5, 2, 3, 4$.

Are these distributions Fat tailed ?

We first examined the GEV family, followed by the GPD, to study the behavior of extreme values. To assess whether a distribution is fat-tailed, we can analyze the asymptotic behavior of the CDFs of well-known heavy-tailed distributions.

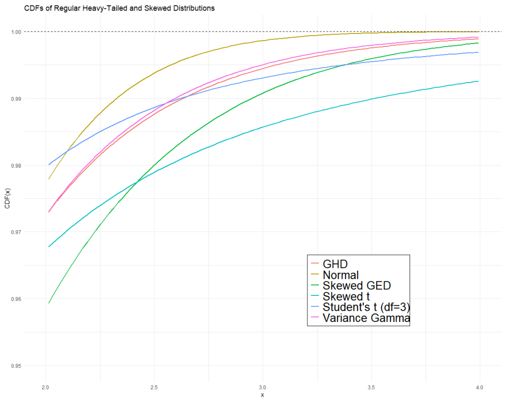

Figure: CDFs of GHD, GED, Student's t, Skewed GED, Skewed Student's t, Normal, and Variance Gamma

Are these distributions Fat tailed ?

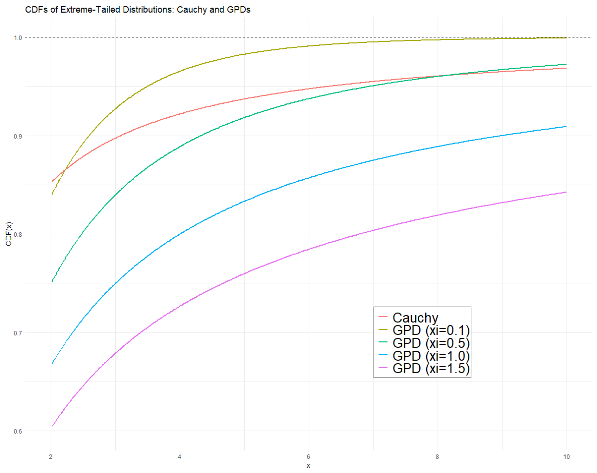

Figure: CDFs of Cauchy, GPD ($\xi = 0.1$), GPD ($\xi = 0.5$), GPD ($\xi = 1$), and GPD ($\xi = 1.5$)

Are these distributions Fat tailed ?

From the figure above, we observe that the distributions commonly used to model financial returns—such as the Student's t, Generalized Error Distribution (GED), Generalized Hyperbolic Distribution (GHD), Variance Gamma, and their skewed variants—are all characterized by fat tails. However, even heavier tails appear in the Generalized Pareto Distribution (GPD) and the Cauchy distribution, as seen from their slower convergence to 1 in the tail probability plots.
In particular, the GPD exhibits significantly slower decay when the shape parameter ξ is high, indicating an extremely heavy tail. This behavior emphasizes the presence of extreme events and supports the importance of explicitly modeling tail risk when assessing the fit of probability distributions to financial return data.

Block Maxima or Peak over Threshold?

We can see a simple comparison between the two approaches, more specifically, using simulated data from ARMA(1,1)-GARCH(1,1)

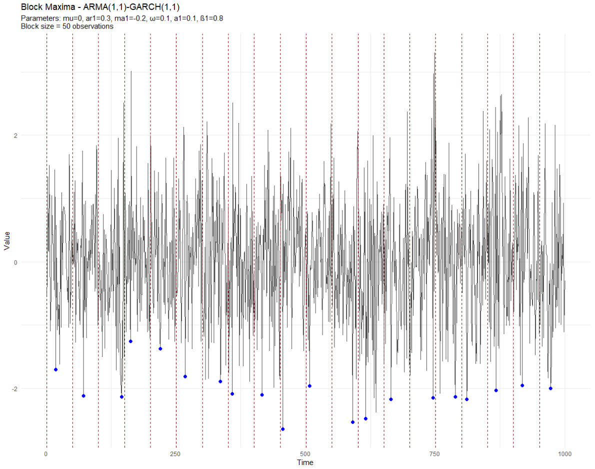

Figure: Block Maxima with block size of 50

Block Maxima or Peaks over Threshold?

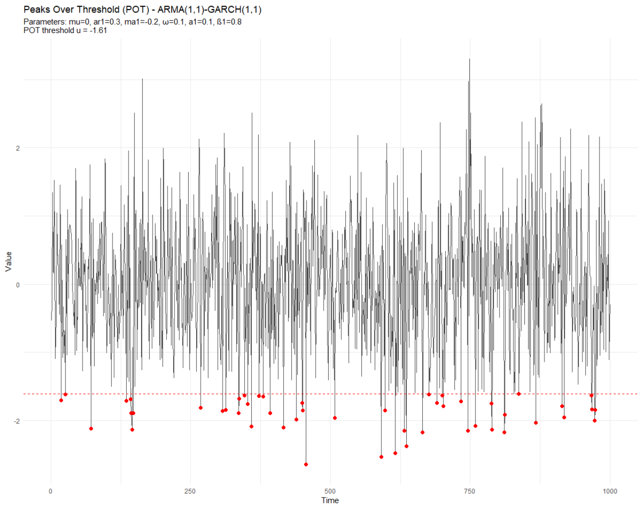

Figure: Peak over Threshold with threshold equal to lower 95% quantile

Block Maxima or Peaks Over Threshold?

As demonstrated by the figures above, this comparison of two methods is conducted within a realistic setting, specifically using an ARMA-GARCH process, a widely accepted model for financial return time series. It is evident that the Block Maxima (BM) method poses challenges for extreme value identification compared to the Peaks Over Threshold (POT) approach. BM requires substantially more data and involves arbitrary decisions on how to partition the data into blocks, which can affect results. Moreover, financial time series often exhibit autocorrelation driven by heteroscedasticity and volatility clustering, leading to clusters of large movements occurring close in time. The BM method, by focusing solely on block maxima and assuming independence, may fail to adequately capture these clusters. For these reasons, the POT method is generally preferred in practice for modeling extremes in financial return series.

Hill Estimator

In the presence of heavy tails, the distribution of exceedances above a high threshold can often be approximated by a Pareto tail:

$$F(x) \approx 1 - Ax^{-1/\xi}, \quad x \to \infty$$

The parameter $\xi > 0$ governs tail heaviness. Larger values of ξ indicate heavier tails; in particular, when $\xi > 1$, even the mean of the distribution becomes infinite.

The Hill estimator is a classical method for estimating the index ξ of a Pareto-type distribution. Given exceedances $x_1, x_2, \ldots, x_{C_T}$ above a threshold u, the Hill estimator is defined as:

$$\hat{\xi} = \frac{1}{C_T} \sum_{i=1}^{C_T} \ln \left(\frac{x_i}{u} \right)$$

where C_T is the number of observations exceeding the threshold u.

Hill Estimator

The Hill estimator is asymptotically efficient when the data are truly Pareto-distributed (Hill, 1975) [11]. However, it is sensitive to the choice of the threshold u or, equivalently, the number of top order statistics k included in the estimate.

Typically, the Hill estimator is visualized via the **Hill plot**, which displays the estimate $\hat{\xi}(k)$ as a function of k. This plot guides the selection of the optimal number of order statistics.

A flat region in the Hill plot suggests a stable and reliable estimate of the tail index ξ. Conversely, rising or falling trends indicate unstable tail behavior or a poor model fit.

Formally, the Hill estimator for a given k is defined as

$$\hat{\xi}(k) = \frac{1}{k} \sum_{i=1}^{k} \ln \left(\frac{x_{(i)}}{x_{(k+1)}} \right)$$

where $x_{(1)} \geq x_{(2)} \geq \cdots \geq x_{(n)}$ are the order statistics of the sample. The Hill plot helps visually identify the region where $\hat{\xi}(k)$ stabilizes, indicating a suitable choice of k.

Hill Estimator

An example of Hill plot on financial data. Alpha indicates $1/\hat{\xi}$.

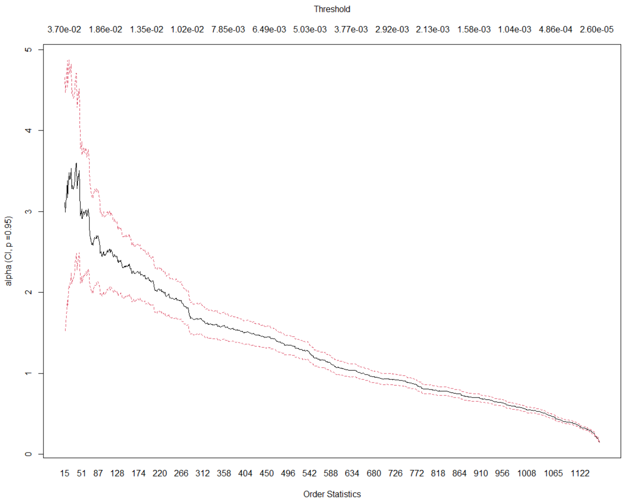

Figure: Hill Plot for S&P 500 data

Fitting Generalized Pareto Distribution

We can now see how to fit the GPD by assuming different approaches for choosing the threshold u.
We start by choosing u equal to a quantile p, so we model the distribution of exceedances above u, that is, $F_u(x)$.
The parameter estimation for the Generalized Pareto Distribution (GPD) can be carried out using different methods. In this context, we consider:

- **Maximum Likelihood Estimation (MLE)**: a classical method that finds the parameters which maximize the likelihood function, assuming the observed exceedances come from a GPD.
- **Probability-Weighted Moments (PWM)**: a method based on matching the theoretical and sample probability-weighted moments, which tends to be more robust than MLE, especially for small sample sizes or in the presence of outliers.

Fitting Generalized Pareto Distribution

R function: gpd_fit

```
gpd_fit <- function(data, threshold,
                    method = c("ml", "pwm")) {
  data <- as.numeric(data)
  exceedances <- data[data > threshold]
  excess <- exceedances - threshold
  Nu <- length(excess)
  xbar <- mean(excess)
  method <- match.arg(method)
  if (method == "ml") {
    s2 <- var(excess)
    xi0 <- -0.5 * (((xbar^2) / s2) - 1)
    beta0 <- 0.5 * xbar * (((xbar^2) / s2) + 1)
    theta_init <- c(xi0, beta0)
    negloglik <- function(theta, tmp) {
      xi <- theta[1]
      beta <- theta[2]
      if (beta <= 0 ||
         (xi <= 0 && max(tmp) > (-beta / xi))) {
        return(1e6)
      } else {
        y <- log1p((xi * tmp) / beta) / xi
        return(length(tmp) * log(beta) +
               (1 + xi) * sum(y))
      }
    }
    fit <- optim(theta_init, negloglik,
                 hessian = TRUE, tmp = excess)
    xi_hat <- fit$par[1]
    beta_hat <- fit$par[2]
    varcov <- solve(fit$hessian)
  }
  if (method == "pwm") {
    a0 <- xbar
    gamma <- -0.35
    delta <- 0
    pvec <- ((1:Nu) + gamma) / (Nu + delta)
    a1 <- mean(sort(excess) * (1 - pvec))
    xi_hat <- 2 - a0 / (a0 - 2 * a1)
    beta_hat <- (2 * a0 * a1) / (a0 - 2 * a1)
    varcov <- NA
  }
  list(threshold = threshold, xi = xi_hat,
       beta = beta_hat, varcov = varcov,
       n_exceed = Nu)
}
```

Fitting Generalized Pareto Distribution

R function: `Fitting GPD to S&P 500 data`

```
# Load Library and Download S&P 500 data
library(quantmod)
getSymbols("^GSPC",from="2015-01-01",to="2025-01-01")

# Compute log returns and obtain the losses
rets=dailyReturn(GSPC,type="log")
retsl=rets*-1
Losses=as.numeric(retsl[retsl>0])

# Fit the distribution assuming the upper 95% quantile as threshold

# Set Threshold
u=quantile(Losses,0.95)

# Fit the distribution
fit1=gpd_fit(data=Losses,threshold = u,method="ml")
# Extract the parameters
fit1$xi
fit1$beta
```

Fitting Generalized Pareto Distribution

Python function: `gpd_fit`

```
def gpd_fit(data, threshold, method="ml"):
    x = np.asarray(data, dtype=float)
    exceedances = x[x > threshold]
    excess = exceedances - threshold
    Nu = excess.size
    xbar = np.mean(excess) if Nu > 0 else np.nan
    method = method.lower()
    if method == "pwm":
        a0 = xbar
        gamma = -0.35
        delta = 0.0
        pvec = (np.arange(1, Nu + 1) + gamma) / (Nu + delta)
        a1 = np.mean(np.sort(excess) * (1.0 - pvec))
        denom = a0 - 2.0 * a1
        xi_hat = 2.0 - a0 / denom
        beta_hat = (2.0 * a0 * a1) / denom
```

```
else:
    s2 = np.var(excess, ddof=1)
    xi0 = -0.5 * (((xbar ** 2) / s2) - 1.0)
    beta0 = 0.5 * xbar * (((xbar ** 2) / s2) + 1.0)
    theta_init = np.asarray([xi0, beta0], float)
    def negloglik(theta, tmp):
        xi, beta = float(theta[0]), float(theta[1])
        if (beta <= 0.0) or (xi <= 0.0 and np.max(tmp)

        > (-beta / xi)):
            return 1e6
        y =
        np.log1p((xi * tmp) / beta) / xi
        return tmp.size *
        np.log(beta) + (1.0 + xi) * np.sum(y)
    res = optimize.minimize(
        fun=negloglik,
        x0=theta_init,
        args=(excess,),
        method="BFGS")
    xi_hat, beta_hat = float(res.x[0]), float(res.x[1])
return {"threshold": float(threshold),"xi":float(xi_hat),
        "beta": float(beta_hat),
        "n_exceed": int(Nu),}
```

Fitting Generalized Pareto Distribution

Python function: `Fitting GPD to S&P 500 data`

```
import numpy as np
import pandas as pd
import scipy as sp
from scipy import optimize
import yfinance as yf

#  Download price data
ticker = '^GSPC'
price = yf.download(ticker, start='2015-01-01', end='2025-01-01',
auto_adjust=True)['Close']

#  Compute log returns
log_ret = np.log(price / price.shift(1)).dropna() * -1 * 100

#  Extract actual losses (positive values of negative returns)
losses = log_ret[log_ret > 0].dropna()

#  Compute 95th percentile of losses (threshold for GPD fit)
losses_95 = np.percentile(losses, 95)

#  Fit the GPD to the losses
gpd_result = gpd_fit(data=losses, threshold=losses_95)
```

Choosing the Threshold u in Peak over Threshold

Selecting an appropriate threshold u is crucial for reliable modeling of exceedances. First of all, balance the bias and variance, indeed, if u is too high, few exceedances lead to high variance in estimates. Whereas, if u is too low, the GPD fit may be poor, causing high bias.

- **Hill Plot:**
  - Estimate the tail index $\hat{\xi}$ over a range of candidate thresholds u.
  - Plot $\hat{\xi}$ against u to identify regions where the estimate stabilizes, indicating a suitable threshold.
- **MSE Minimization and Automated Methods:**
  - Select u by minimizing the mean squared error (MSE) of parameter estimates.
  - Use automated algorithms to detect stable regions in the Hill plot where $\hat{\xi}$ remains consistent.
- **Practical Considerations:**
  - Initial choice of u is often informed by domain knowledge and expertise.
  - Automated methods should be validated using diagnostic plots.
  - Performing sensitivity analysis on u ensures the robustness of risk measures such as VaR or Expected Shortfall.

Threshold Selection

The selection of the threshold u is a fundamental step in Extreme Value Theory (EVT), as it determines the balance between bias and variance when estimating the tail behavior of the distribution.

In this analysis, we rely exclusively on some automated approaches:

1. The Mean Absolute Deviation (MAD) Distance Method
2. The Eye-Ball Method
3. Double Bootstrap Method
4. Path Stability Method

These methods allow for a data-driven identification of the threshold based on stability and goodness-of-fit of the tail.

Mean Absolute Deviation Distance Metric

This method, proposed by Danielsson et al. [7], aims to find the number of extreme order statistics k such that the empirical tail closely aligns with a fitted Pareto tail.
For each candidate k, we compute the Hill estimator $\hat{\xi}_k$ and measure the distance between empirical and theoretical quantiles. The optimal value is

$$k_0 = \arg\min_k Q(k)$$

where $Q(k)$ is the mean absolute deviation:

$$Q_n^{\text{MAD}} = \frac{1}{T} \sum_{j=1}^{T} \left| r_{n-j,n} - q(j,k) \right|.$$

The theoretical quantile is approximated by

$$q(j,k) = r_{n-j+1,n} \left(\frac{k}{j} \right)^{1/\hat{\xi}_k},$$

where $\hat{\xi}_k$ is the Hill estimator computed from the top k order statistics. The method seeks the value of k that minimizes the average deviation between these estimated quantiles and the observed exceedances.

Mean Absolute Deviation Distance Metric

R function: `Mean Absolute Deviation Distance Metric`

```r
mad_threshold <- function(data, ts = 0.15) {
  # Sort data in decreasing order
  xstat <- sort(data, decreasing = TRUE)
  n <- length(xstat)
  T <- floor(n * ts)
  i <- 1:(n - 1)

  # Compute inverse Hill estimator
  h <- (cumsum(log(xstat[i])) / i) - log(xstat[i + 1])

  # Sort data in increasing order for quantile comparison
  xstat <- sort(data)

  # Matrix to store absolute deviations
  A <- matrix(ncol = T - 1, nrow = T - 1)

  for (k in 1:(T - 1)) {
    for (j in 1:(T - 1)) {
      A[k, j] <- abs((((k / j) * xstat[n - k + 1]^(1 / h[k]))^h[k]) - xstat[n - j])
    }
  }

  # Compute row means (MAD) and find optimal k
  M <- rowMeans(A)
  kstar <- which.min(M)
  u <- rev(xstat)[kstar]

  return(list(
    k0 = kstar,
    threshold = u,
    tail.index = 1 / h[kstar]
  ))
} # here the tail index is equal to 1/xi, k0, the number of observations greater than threshold.
```

Eye-Ball Method

The Eye-Ball method is a heuristic approach that identifies a stable region (plateau) in the sequence of Hill estimates. It is a method defined by Danielsson et al. [7]. A window of size w slides along the Hill plot. Within each window, we check whether the estimates remain within a tolerance band ϵ around the first value. If a sufficient proportion h of points satisfy this condition, the corresponding k is accepted:

$$k^* = \min \left\{ k \mid \frac{1}{w} \sum_{i=1}^{w} \mathbb{I}\left(\hat{\xi}(k+i) \in \hat{\xi}(k) \pm \epsilon \right) > h \right\}.$$

Typically, $w \approx 0.01n$, $h = 90\%$, and $\epsilon = 0.3$.

Eye-Ball Method

R function: `Eye-Ball Method`

```
eye <-function(data,ws=0.01,epsilon=0.3,h=0.9){

n=length(data)
w=floor(ws*n)

i=1:(n-1)
x=sort(data,decreasing=TRUE)
gamma=(cumsum(log(x[i]))/i)-log(x[i+1])
alpha=1/gamma

count=0;erg=c()
for (k in 2:(length(alpha)-w)){
  for (i in 1:length(w)){
    if (alpha[k+i]<(alpha[k]+epsilon) && alpha[k+i]>(alpha[k]-epsilon)) {
      count=count+1
  } else {
    count=count
  }
  }
  erg[k]=count/w
}
erg=erg>h
k0=min(which(erg==1))
u=x[k0]
ti=alpha[k0]
list=list(k0=k0,threshold=u,tail.index=ti)
list
}
```

Eye-Ball Method

Python function: `Eye-Ball Method`

```
def eye(data, ws=0.01, epsilon=0.3, h=0.9):
    data = data.iloc[:, 0]
    data = data[data > 0]
    data = np.asarray(data, dtype=float)
    n = len(data)
    w = int(np.floor(ws * n))
    i = np.arange(1, n)
    x = np.sort(data)[::-1]  # decrescente

    gamma = (np.cumsum(np.log(x[:-1])) / i) - np.log(x[1:])
    alpha = 1 / gamma

    erg = np.zeros_like(alpha)
    for k in range(2, len(alpha) - w):
        count = 0
        for j in range(1, w + 1):  # equivale a R: i in 1:length(w), con length(w) = w
            if (alpha[k + j] < alpha[k] + epsilon) and (alpha[k + j] > alpha[k] - epsilon):
                count += 1
        erg[k] = count / w

    erg_bool = erg > h

    if np.any(erg_bool):
        k0 = np.argmax(erg_bool)
        u = x[k0]
        ti = alpha[k0]
    else:
        k0 = None
        u = None
        ti = None

    return {'k0': k0, 'threshold': u, 'tail.index': ti}
```

Double Bootstrap Approach

Another popular approach is the double bootstrap described by Danielsson (2001) [6] which provides a data-driven, objective approach to selecting the optimal threshold k_0.
Purpose: Minimize the Asymptotic Mean Squared Error (AMSE) of the tail index estimator.

Asymptotic Mean Squared Error

$$\text{AMSE}(n, \kappa) = \mathbb{E}[(\hat{\xi}_{n,\kappa} - \xi)^2]$$

Problem: ξ is unknown, so AMSE can't be computed directly.
Solution: Use a second estimator as proxy:

$$\text{AMSE}(n, \kappa) = \mathbb{E}[(\hat{\xi}^{(2)}_{n_1,\kappa} - \hat{\xi}^{(1)}_{n_2,\kappa})^2]$$

- $\hat{\xi}^{(1)}$: first moment (Hill)
- $\hat{\xi}^{(2)}$: second moment

Double Bootstrap Approach

The estimators can be written using log-excesses over order statistics:

Moment Definitions

$$M_\kappa^{(1)} = \frac{1}{\kappa}\sum_{i=1}^{\kappa} \log\left(\frac{x_i}{x_{\kappa+1}}\right)$$

$$M_\kappa^{(2)} = \frac{1}{\kappa}\sum_{i=1}^{\kappa} \left(\log\left(\frac{x_i}{x_{\kappa+1}}\right)\right)^2$$

So that the adjusted estimator becomes:

$$Q_n(\kappa) = M_\kappa^{(2)} - 2(M_\kappa^{(1)})^2$$

Objective: For each κ, compute:

$$\widehat{\mathrm{AMSE}}(\kappa) = \mathbb{E}[(Q_n(\kappa))^2]$$

Then find the κ that minimizes this quantity.

Double Bootstrap Approach

The steps are the following:

1. Generate B bootstrap samples of size $n_1 = \lfloor n^{\epsilon} \rfloor$
2. For each bootstrap sample and for each $\kappa = 1, \dots, n_1 - 1$, compute $Q_n^{(1)}(\kappa)$
3. Repeat with a second bootstrap sample of size $n_2 = \lfloor \frac{n_1^2}{n} \rfloor$ to compute $Q_n^{(2)}(\kappa)$
4. Compute mean squared errors: $\widehat{\text{AMSE}}^{(1)}(\kappa), \widehat{\text{AMSE}}^{(2)}(\kappa)$
5. Find $\kappa_1^\star = \arg\min \widehat{\text{AMSE}}^{(1)}(\kappa)$ and $\kappa_2^\star = \arg\min \widehat{\text{AMSE}}^{(2)}(\kappa)$

Double Bootstrap Approach

Once $k_1^\star$ and $k_2^\star$ are found:

Threshold Selection Rule

$$k_0^\star = \left\lfloor \frac{k_1^{\star 2}}{k_2^\star} \left(\frac{Z}{N} \right)^{\text{Exp}} \right\rfloor + 1$$

$$\text{Exp} = \frac{\log n_1 - \log k_1^\star}{\log n_1}$$

$$Z = \log^2 k_1^\star$$

$$N = (2 \log n_1 - \log k_1^\star)^2$$

Then, the tail index is estimated via the Hill estimator:

$$\hat{\xi}_{k_0^\star} = \frac{1}{k_0^\star} \sum_{i=1}^{k_0^\star} \log \left(\frac{x_i}{x_{k_0^\star + 1}} \right)$$

Double Bootstrap Approach

R function: `Double Bootstrap`

```
DoubleBoot <- function(data, B = 500, epsilon = 0.9) {
  n <- length(data)
  n1 <- floor(n^epsilon)
  n2 <- floor(n1^2 / n)

  Qn <- function(x, k) {
    x <- sort(x, decreasing = TRUE)
    l1 <- log(x[1:k]) - log(x[k + 1])
    mean(l1^2) - 2 * mean(l1)^2
  }

  qn1 <- matrix(NA, nrow = B, ncol = n1 - 1)
  qn2 <- matrix(NA, nrow = B, ncol = n2 - 1)

  for (b in 1:B) {
    x1 <- sample(data, n1, replace = TRUE)
    x2 <- sample(data, n2, replace = TRUE)
    qn1[b, ] <- sapply(1:(n1 - 1), function(k) Qn(x1, k))
    qn2[b, ] <- sapply(1:(n2 - 1), function(k) Qn(x2, k))
  }

  qn1star <- colMeans(qn1^2)
  qn2star <- colMeans(qn2^2)

   k1star <- which.min(qn1star)
  k2star <- which.min(qn2star)

  Exp <- (log(n1) - log(k1star)) / log(n1)
  Z <- log(k1star)^2
  N <- (2 * log(n1) - log(k1star))^2
  k0star <- floor((k1star^2 / k2star) * (Z / N)^Exp) + 1

  x_sorted <- sort(data, decreasing = TRUE)
  u <- x_sorted[k0star]
  rho <- log(k1star) / (-2 * log(n1) + 2 * log(k1star))

  helphill <- function(k) {
    x <- sort(data, decreasing = TRUE)
    mean(log(x[1:k]) - log(x[k + 1]))
  }

  ti <- 1 / helphill(k0star)

  list(
    sec.order.par = rho,
    k0 = k0star,
    threshold = u,
    tail.index = ti)
}
```

Double Bootstrap Approach

Python function: `Double Bootstrap`

```
def DoubleBoot(data, B=500, epsilon=0.9):
    if isinstance(data, pd.DataFrame):
        data = data.iloc[:, 0]
    data = data[data > 0]
    data = np.asarray(data, dtype=float)

    n = len(data)
    n1 = int(np.floor(n ** epsilon))
    n2 = int(np.floor((n1 ** 2) / n))

    def Qn(x, k):
        x_sorted = np.sort(x)[::-1]
        logs = np.log(x_sorted[:k]) - np.log(x_sorted[k])
        return np.mean(logs ** 2) - 2 * (np.mean(logs) ** 2)

    qn1 = np.zeros((B, n1 - 1))
    qn2 = np.zeros((B, n2 - 1))
    for b in range(B):
        x1 = np.random.choice(data, n1, replace=True)
        x2 = np.random.choice(data, n2, replace=True)
        qn1[b, :] = [Qn(x1, k) for k in range(1, n1)]
        qn2[b, :] = [Qn(x2, k) for k in range(1, n2)]

    qn1star = np.mean(qn1 ** 2, axis=0)
    qn2star = np.mean(qn2 ** 2, axis=0)
```

```
k1star = int(np.argmin(qn1star)) + 1
k2star = int(np.argmin(qn2star)) + 1

Exp = (np.log(n1) - np.log(k1star)) / np.log(n1)
Z = np.log(k1star) ** 2
N = (2 * np.log(n1) - np.log(k1star)) ** 2
k0star = int(np.floor((k1star ** 2 / k2star)
* (Z / N) ** Exp)) + 1

x_sorted = np.sort(data)[::-1]
u = x_sorted[k0star]
rho = np.log(k1star) / (-2 * np.log(n1)
+ 2 * np.log(k1star))

def helphill(k):
    x_sorted = np.sort(data)[::-1]
    logs = np.log(x_sorted[:k]) - np.log(x_sorted[k])
    return np.mean(logs)

ti = 1 / helphill(k0star)
return {
    'sec.order.par': rho,
    'k0': k0star,
    'threshold': u,
    'tail.index': ti}
```

Path Stability Method

We recall that, tail index estimation using the Hill estimator is highly sensitive to the threshold parameter k. An interesting alternative is presented by the Path Stability Method (Caeiro and Gomes, 2016, and Gomes 2011), [4], [10], [9], it offers a heuristic and data-driven rule to identify a "stable" region of the Hill plot. The method is designed to locate the optimal number of order statistics k_0 where the Hill estimator stabilizes

Underlying idea:

- Consider the Hill plot: $\hat{\xi}(k) = \hat{\xi}_{H,k,n}$ for $k = 1, 2, \ldots, n-1$
- Identify the flattest segment (most stable path) of the Hill curve
- Use a **rounding strategy** to detect plateaus in the curve
- Final tail index estimate is taken at the most persistent stable level

Path Stability Method

The steps are the following: **Step 1:** Compute the Hill estimator for all $k = 1, \dots, n-1$:

$$\hat{\xi}(k) = \frac{1}{k} \sum_{i=1}^{k} \log \left(\frac{x_i}{x_{k+1}} \right)$$

Step 2: Find the minimum number of decimal places j_0 such that the rounded values of $\hat{\xi}(k)$ become distinct:

$$a_k^{(\hat{\xi})}(j) = \text{round}(\hat{\xi}(k), j)$$

Step 3: Identify the longest sequence of equal consecutive values in $a_k^{(\hat{\xi})}(j_0)$.

- Let $k_{\min}^{(\hat{\xi})}$ and $k_{\max}^{(\hat{\xi})}$ be the bounds of this stable region.
- Set run length: $\ell_{\hat{\xi}} = k_{\max}^{(\hat{\xi})} - k_{\min}^{(\hat{\xi})}$

Path Stability Method

Step 4: Increase resolution using $j_0 + 2$ decimal places:

$$\hat{\xi}(k) = a_k^{(\hat{\xi})}(j_0 + 2), \quad \text{for } k \in [k_{\min}^{(\hat{\xi})}, k_{\max}^{(\hat{\xi})}]$$

- Find the **mode** of these values.
- Denote $K_{\hat{\xi}}$ as the set of k where the mode occurs.

Step 5: Define:

$$\hat{k}_{\hat{\xi}} = \max(K_{\hat{\xi}})$$

Step 6: Compute the final tail index estimate:

$$\hat{\xi}_{H|PS} = \hat{\xi}(\hat{k}_{\hat{\xi}})$$

Path Stability Method

R function: Path Stability

```
PS <-
function(data,j=1){
n=length(data)
x1=sort(data, decreasing=TRUE)
i=1:(n-1)
h=(cumsum(log(x1[i]))/i)-log(x1[i+1])

help=round(h,j)
l=rle(help)$lengths
Max=which(l==max(l))
kmin=sum(l[1:(Max-1)])+1
kmax=kmin+max(l)-1
run=kmax-kmin

K=round(h[kmin:kmax],j+2)

Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
if (sum(duplicated(K))==0) {
  modus=K[length(K)]
} else{
  modus=Mode(K)
}

k0star=max(which(round(h,j+2)==modus))
u=x1[k0star]
ti=1/h[k0star]
list=list(k0=k0star,threshold=u,tail.index=ti)
list
}
```

Path Stability Method

Python function: `Path Stability`

```
def PS(data, j=1):
    if isinstance(data, pd.DataFrame):
        data = data.iloc[:, 0]
    data = data[data > 0]
    data = np.asarray(data, dtype=float)
    n = len(data)
    x1 = np.sort(data)[::-1]
    i = np.arange(1, n)
    h = (np.cumsum(np.log(x1[:-1])) / i) - np.log(x1[1:])
    h_rounded = np.round(h, decimals=j)
    run_lengths = []
    current = h_rounded[0]
    count = 1
    for val in h_rounded[1:]:
        if val == current:
            count += 1
        else:
            run_lengths.append((current, count))
            current = val
            count = 1
    run_lengths.append((current, count))
```

```
lengths = [length for value, length in run_lengths]
max_run_index = np.argmax(lengths)
kmin = sum(lengths[:max_run_index])
kmax = kmin + lengths[max_run_index] - 1
run = kmax - kmin
K = np.round(h[kmin:kmax + 1], decimals=j + 2)
if len(set(K)) == len(K):
    modus = K[-1]
else:
    modus = mode(K)
h_rounded_j2 = np.round(h, decimals=j + 2)
matching_indices = np.where(h_rounded_j2 == modus)[0]
if len(matching_indices) == 0:
    return {'k0': None, 'threshold':
    None, 'tail.index': None}
k0star = int(np.max(matching_indices))
u = x1[k0star]
tail_index = 1 / h[k0star]
return {
    'k0': k0star + 1,
    'threshold': u,
    'tail.index': tail_index
}
```

References I

Balkema, A. A., & de Haan, L. (1974). Residual life time at great age. *The Annals of Probability*, *2*(5), 792–804.

Brooks, C. (2019). *Introductory Econometrics for Finance* (4th ed.). Cambridge University Press.

Campbell, J. Y., Lo, A. W., MacKinlay, A. C., & Whitelaw, R. F. (1998). *The Econometrics of Financial Markets*. Princeton University Press.

Caeiro, J., & Gomes, M. I. (2016). Threshold selection in extreme value analysis. In D. K. Dey & J. Yan (Eds.), *Extreme Value Modeling and Risk Analysis: Methods and Applications* (pp. 69–86). Chapman and Hall/CRC.

Danielsson, J. (2011). Financial risk forecasting: The theory and practice of forecasting market risk with implementation in R and Matlab. Wiley.

Danielsson, J., de Haan, L., Peng, L., & de Vries, C. G. (2001). Using a bootstrap method to choose the sample fraction in tail index estimation. *Journal of Multivariate Analysis*, *76*(2), 226–248. https://doi.org/10.1006/jmva.2000.1924

References II

Danielsson, J., Ergun, L. M., de Haan, L., & de Vries, C. G. (2016). Tail index estimation: Quantile driven threshold selection. *Review of Economics and Statistics*, *98*(5), 861–875. https://doi.org/10.1162/REST_a_00584

Gnedenko, B. (1943). Sur la distribution limite du terme maximum d'une série aléatoire. *Annals of Mathematics*, *44*(3), 423–453.

Gomes, M. I., Henriques-Rodrigues, L., & Miranda, M. C. (2011). Reduced-bias location-invariant extreme value index estimation: A simulation study. *Communications in Statistics - Simulation and Computation*, *40*(3), 424–447. https://doi.org/10.1080/03610918.2011.554590

Gomes, M. I., Henriques-Rodrigues, L., Fraga Alves, M. I., & Manjunath, B. (2013). Adaptive PORT-MVRB estimation: An empirical comparison of two heuristic algorithms. *Journal of Statistical Computation and Simulation*, *83*(6), 1129–1144. https://doi.org/10.1080/00949655.2011.647495

Hill, B. M. (1975). A simple general approach to inference about the tail of a distribution. *The Annals of Statistics*, *3*(5), 1163–1174.

References III

Pickands III, J. (1975). Statistical inference using extreme order statistics. *The Annals of Statistics*, *3*(1), 119–131.

Taleb, N. N. (2020). *Statistical consequences of fat tails: Real world preasymptotics, epistemology, and applications*. arXiv preprint arXiv:2001.10488.

Ossberger, J. (2020). tea: Threshold Estimation Approaches (Version 1.1) [R package]. Comprehensive R Archive Network (CRAN). https://CRAN.R-project.org/package=tea

Pfaff, B., Zivot, E., McNeil, A., & Stephenson, A. (2018). evir: Extreme Values in R (Version 1.7-4) [R package]. Comprehensive R Archive Network (CRAN). https://CRAN.R-project.org/package=evir