Returns Distribution Fitting Pipeline

Complete Documentation

Version 2.0 | November 2025

Table of Contents

1. Overview
2. Installation
3. Quick Start
4. Configuration Parameters
5. Methodology
6. Output Files
7. Interpreting Results
8. Advanced Usage
9. Programmatic API
10. Performance Optimization
11. Troubleshooting
12. Best Practices
13. References

1. Overview

The Returns Distribution Fitting Pipeline is a comprehensive Python tool for analyzing financial returns distributions using both parametric and non-parametric methods. It automates the entire workflow from data cleaning to statistical testing and visualization.

Key Features:

• Robust price data cleaning with support for multiple formats

• Flexible filtering by ticker symbol and date ranges

• Comprehensive distribution scanning (all scipy.stats infinite-support distributions)

• Non-parametric KDE with configurable bandwidth selection

• Kolmogorov-Smirnov goodness-of-fit testing

• Side-by-side visualization of best parametric vs. KDE fits

• Detailed CSV reports with full distribution rankings

2. Installation

Required Dependencies:

```
pip install pandas numpy scipy matplotlib KDEpy tqdm
```

Optional for Parquet support:

```
pip install pyarrow
```

3. Quick Start

Basic command-line usage:

```
python returns_dist_fit_v2.py --config returns_dist_fit_v2_config.json
```

Example configuration file:

```
{ "input": "master_stock.csv", "output": null, "kde_bw": "ISJ", "ticker":
"AAPL", "start_date": null, "end_date": null, "distributions": ["norm", "t",
"nct", "laplace"], "alpha": 0.01, "quiet": false }
```

4. Configuration Parameters

Parameter	Type	Description	Default
input	string	Path to CSV/Parquet data file	Required
output	string	Output PNG filename (null=auto)	null
ticker	string	Ticker symbol to filter	null

start_date	string	Start date (YYYY-MM-DD)	null
end_date	string	End date (YYYY-MM-DD)	null
kde_bw	string/float	KDE bandwidth method	"ISJ"
distributions	list	List of distributions to test	null (all)
alpha	float	KS test significance level	0.01
quiet	boolean	Suppress verbose output	false

5. Methodology

5.1 Price Data Cleaning

The pipeline applies multi-stage normalization to handle various price formats:

• Convert parentheses to negatives: (1,500.25) → -1500.25

• Remove thousands separators: 41,929.00 → 41929.00

• Strip currency symbols and non-numeric characters

5.2 Returns Calculation

Simple returns are computed grouped by ticker symbol:

```
Returns[t] = (Price[t] - Price[t-1]) / Price[t-1]
```

5.3 Kernel Density Estimation

FFT-based KDE with Gaussian kernel provides non-parametric density estimates. Bandwidth selection methods include Improved Sheather-Jones (ISJ) and Silverman's rule.

5.4 Parametric Fitting

Maximum Likelihood Estimation (MLE) via scipy.stats fits each distribution with automatic parameter bounds and convergence handling.

5.5 Kolmogorov-Smirnov Testing

The KS test compares the empirical CDF with each theoretical CDF. The test statistic D measures maximum deviation; p-value indicates goodness-of-fit. Higher p-values indicate better fit.

6. Output Files

6.1 Visualization: {ticker}_returns_fit.png

Side-by-side comparison plot with 300 DPI resolution showing:

• Left panel: Best parametric distribution overlaid on histogram

• Right panel: KDE (non-parametric) overlaid on histogram

6.2 Statistical Report: {ticker}_returns_fit_ks_scan.csv

CSV file with KS test results for all distributions, sorted by p-value (descending):

Column	Description
distribution	Distribution name or "KDE"
D_stat	Kolmogorov-Smirnov test statistic
p_value	KS test p-value (higher = better fit)
decision_alpha	"Fail to reject" or "Reject" at alpha
alpha	Significance level used for decision
parameters	Fitted parameters (tuple or bandwidth)

7. Interpreting Results

7.1 P-Value Interpretation

P-Value Range	Interpretation	Action
p > 0.05	Good fit (5% level)	Accept distribution
p > 0.01	Very good fit (1% level)	Strong evidence for fit
0.001 < p < 0.01	Marginal fit	Consider alternatives
p < 0.001	Poor fit	Reject distribution

7.2 D-Statistic Interpretation

D-Statistic	Quality of Fit
D < 0.05	Excellent
0.05 ≤ D < 0.10	Good
0.10 ≤ D < 0.20	Acceptable
D ≥ 0.20	Poor

8. Advanced Usage

8.1 Batch Processing Multiple Tickers

```
for ticker in AAPL NVDA TSLA MSFT; do python returns_dist_fit_v2.py --input data.csv --ticker $ticker done
```

8.2 Custom Distribution Sets

Focus on specific distribution families by providing a custom list:

```
{"distributions": ["norm", "t", "nct", "skewnorm", "cauchy"]}
```

9. Programmatic API

```
import pandas as pd from returns_dist_fit_v2 import analyze_returns df = pd.read_csv('data.csv') results = analyze_returns( df=df, ticker='AAPL', kde_bw='ISJ', alpha=0.01, verbose=True ) print(f"Best: {results['best_distribution']}") print(results['ks_scan'])
```

10. Performance Optimization

For large datasets, consider:

• Subsample data for initial exploration

• Limit distribution list to most relevant

• Use Parquet format for faster I/O

• Enable quiet mode for batch processing

11. Best Practices

1. Always visually inspect plots; don't rely solely on p-values
2. Use at least 200-500 observations for reliable fitting
3. Compare distributions across different market regimes
4. ISJ bandwidth generally outperforms Silverman's rule
5. Use α=0.01 for conservative tests, α=0.05 for standard testing
6. Cross-validate fitted distributions on out-of-sample data

12. References

Massey, F. J. (1951). The Kolmogorov-Smirnov Test for Goodness of Fit. Journal of the American Statistical Association.

Sheather, S. J., & Jones, M. C. (1991). A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation.

Johnson, N. L., Kotz, S., & Balakrishnan, N. (1995). Continuous Univariate Distributions, Volume 2. Wiley.

SciPy Documentation: scipy.stats module. https://docs.scipy.org/doc/scipy/reference/stats.html

KDEpy Documentation: FFT-based Kernel Density Estimation. https://kdepy.readthedocs.io/

For support: swirlbeast@gmail.com