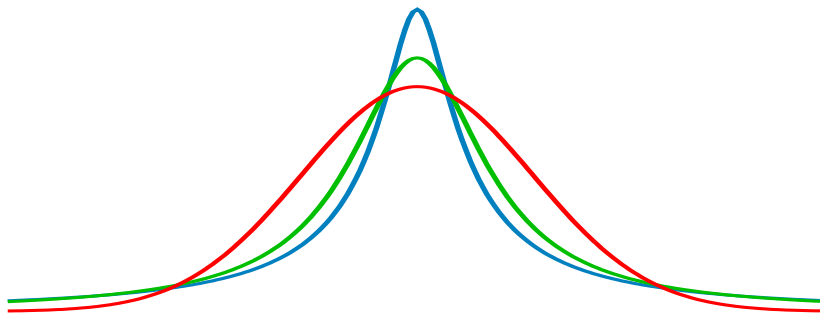


User Manual for STABLE 5.3

R Version



Abstract

This manual gives information about the STABLE library, which computes basic quantities for univariate stable distributions: densities, cumulative distribution functions, quantiles, and simulation. Statistical routines are given for fitting stable distributions to data and assessing the fit. Utility routines give information about the program and perform related calculations. Quick spline approximations of the basic functions are provided. Densities, cumulative distribution functions and simulation for discrete/quantized stable distributions are described.

The multivariate module gives functions to compute bivariate stable densities, simulate stable random vectors, and fit bivariate stable data. In the radially symmetric case, the amplitude densities, cumulative distribution functions, quantiles are computed for dimension up to 100.

©2002-2016 by Robust Analysis, Inc.
www.RobustAnalysis.com
info@robustanalysis.com
Processed July 18, 2017

Contents

1	Univariate Stable Introduction	4
2	Univariate Stable Functions	5
2.1	Basic functions	5
2.1.1	Test scripts	5
2.1.2	Stable densities	5
2.1.3	Stable distribution functions	6
2.1.4	Stable quantiles	6
2.1.5	Simulate stable random variates	6
2.1.6	Stable hazard function	6
2.1.7	Derivative of stable densities	6
2.1.8	Second derivative of stable densities	6
2.1.9	Stable score/nonlinear function	6
2.2	Statistical functions	7
2.2.1	Estimating stable parameters	7
2.2.2	Maximum likelihood estimation	7
2.2.3	Maximum likelihood estimation with restricted parameters	7
2.2.4	Maximum likelihood estimation with search control	7
2.2.5	Quantile based estimation	8
2.2.6	Empirical characteristic function estimation	8
2.2.7	Fractional moment estimation	8
2.2.8	Log absolute moment estimation	8
2.2.9	Quantile based estimation, version 2	8
2.2.10	U statistic based estimation	8
2.2.11	Confidence intervals for ML estimation	9
2.2.12	Information matrix for stable parameters	9
2.2.13	Log-likelihood computation	9
2.2.14	Chi-squared goodness-of-fit test	9
2.2.15	Kolmogorov-Smirnov goodness-of-fit test	9
2.2.16	Likelihood ratio test	10
2.2.17	Stable regression	10
2.3	Informational/utility functions	11
2.3.1	Version information	11
2.3.2	Modes of stable distributions	11
2.3.3	Set internal tolerance	12
2.3.4	Get internal tolerance	12
2.3.5	Convert between parameterizations	12
2.3.6	Omega function	12
2.4	Series approximations to basic distribution functions	12
2.4.1	Series approximation of stable pdf around the origin	13
2.4.2	Series approximation of stable cdf around the origin	13
2.4.3	Series approximation of stable pdf at the tail	13
2.4.4	Series approximation of stable cdf at the tail	13
2.5	Faster approximations to basic functions	13
2.5.1	Quick stable density computation	13
2.5.2	Quick stable cumulative computation	14
2.5.3	Quick stable log pdf computation	14
2.5.4	Quick stable quantile computation	14
2.5.5	Quick stable hazard function computation	14
2.5.6	Quick stable likelihood computation	14
2.5.7	Quick stable score/nonlinear function	14
2.6	Discrete stable distributions	14

2.6.1	Discrete stable density	15
2.6.2	Quick discrete stable density	15
2.6.3	Discrete stable cumulative distribution function	15
2.6.4	Quick discrete stable cumulative distribution function	15
2.6.5	Simulate discrete stable random variates	15
2.6.6	Simulate discrete stable random variates with specified saturation probability	15
2.6.7	Find scale γ to have a specified saturation probability for a discrete stable distribution	15
2.6.8	Discrete maximum likelihood estimation	15
3	Multivariate Stable Introduction	17
4	Multivariate Stable Functions	18
4.1	Define multivariate stable distribution	18
4.1.1	Independent components	19
4.1.2	Isotropic stable	19
4.1.3	Elliptical stable	19
4.1.4	Discrete spectral measure	19
4.1.5	Discrete spectral measure in 2 dimensions	20
4.1.6	Undefine a stable distribution	20
4.2	Basic functions	20
4.2.1	Density function	20
4.2.2	Cumulative function	21
4.2.3	Cumulative function (Monte Carlo)	21
4.2.4	Multivariate simulation	21
4.2.5	Find a 2-dimensional rectangle with probability at least p	21
4.3	Statistical functions	21
4.3.1	Estimate a discrete spectral measure - fit a stable distribution to bivariate data	21
4.3.2	Estimate parameter functions	22
4.3.3	Fit an elliptical stable distribution to multivariate data	22
4.4	Amplitude distribution	22
4.4.1	Amplitude cumulative distribution function	22
4.4.2	Amplitude density	22
4.4.3	Amplitude quantiles	22
4.4.4	Simulate amplitude distribution	23
4.4.5	Fit amplitude data	23
4.4.6	Amplitude nonlinear function	23
4.5	Faster approximations to multivariate routines	23
4.5.1	Quick log-likelihood for bivariate isotropic case	23
4.5.2	Quick amplitude density in bivariate case	23
4.6	Bivariate discrete stable distribution	24
4.6.1	Discrete bivariate density	24
4.7	Multivariate informational/utility functions	24
4.7.1	Information about a distribution	24
4.7.2	Compute projection parameter functions	24
4.7.3	Multivariate convert parameterization	24
5	Error/return codes	25
	References	27
	Index	27

1 Univariate Stable Introduction

Stable distributions are a class of probability distributions that generalize the normal distribution. Stable distributions are a four parameter family: α is the tail index, or index of stability, and is in the range $0 < \alpha \leq 2$, β is a skewness parameter and is in the range $-1 \leq \beta \leq 1$, γ is a scale parameter and must be positive, and δ is a location parameter, an arbitrary real number.

Since there are no formulas for the density and distribution function of a general stable law, they are described in terms of their characteristic function (see below). The main purpose of the STABLE program is to make these distributions accessible in practical problems. The package enables the calculation of stable densities, cumulative distribution functions, quantiles, etc. It also can fit data by several different estimation procedures.

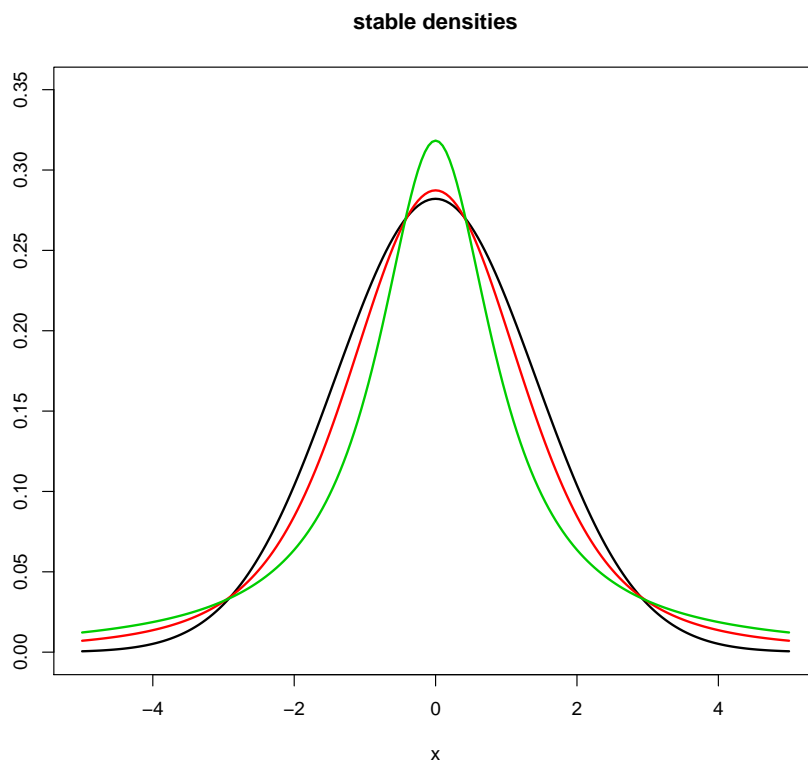


Figure 1: Symmetric stable densities ($\beta = 0$) with $\alpha = 2$ (Gaussian, in black), $\alpha = 1.5$ (red), and $\alpha = 1$ (Cauchy, green).

There are numerous meanings for these parameters. We will focus on two here, which we call the 0-parameterization and the 1-parameterization. The STABLE programs use a variable `param` to specify which of these parameterizations to use. If you are only concerned with symmetric stable distributions, the two parameterizations are identical. For non-symmetric stable distributions, we recommend using the 0-parameterization for most statistical problems, and only using the 1-parameterization in special cases, e. g. the one sided distributions when $\alpha < 1$ and $\beta = \pm 1$.

A random variable X is $S(\alpha, \beta, \gamma, \delta; 0)$ if it has characteristic function (Fourier transform)

$$E \exp(iuX) = \begin{cases} \exp(-\gamma^\alpha |u|^\alpha [1 + i\beta(\tan \frac{\pi\alpha}{2})(\text{sign } u)(|\gamma u|^{1-\alpha} - 1)] + i\delta u) & \alpha \neq 1 \\ \exp(-\gamma |u| [1 + i\beta \frac{2}{\pi}(\text{sign } u) \ln(\gamma |u|)] + i\delta u) & \alpha = 1. \end{cases} \quad (1)$$

A random variable X is $S(\alpha, \beta, \gamma, \delta; 1)$ if it has characteristic function

$$E \exp(iuX) = \begin{cases} \exp(-\gamma^\alpha |u|^\alpha [1 - i\beta(\tan \frac{\pi\alpha}{2})(\text{sign } u)] + i\delta u) & \alpha \neq 1 \\ \exp(-\gamma|u| [1 + i\beta\frac{2}{\pi}(\text{sign } u) \ln |u|] + i\delta u) & \alpha = 1. \end{cases} \quad (2)$$

Note that if $\beta = 0$, then these two parameterizations are identical, it is only when $\beta \neq 0$ that the asymmetry term (the imaginary factor involving $\tan \frac{\pi\alpha}{2}$ or $\frac{2}{\pi}$) becomes relevant. More information on parameterizations and about stable distributions in general can be found at <http://academic2.american.edu/~jpnolan>, which has a draft of the first chapter of Nolan (2017).

The next section gives a description of the basic univariate functions in STABLE.

2 Univariate Stable Functions

Interfaced STABLE functions require input variables, and return the results of the computations. The interface computes the lengths of all arrays, specifies default values for some of the variables in some case, and handles return codes and results.

The parameters of the stable distribution must be specified. The R interface requires `alpha`, but lets the user leave off other arguments, in which case `beta` defaults to be 0, `gamma` defaults to be 1, and `delta` defaults to be 0. `param` defaults to 0 in all cases. In some of the utility or statistical routines, the 4 stable parameters are passed in a vector `theta=(alpha,beta,gamma,delta)`. In these cases, R requires all 4 parameters to be specified.

The STABLE interface prints an error message when an error occurs. If an error occurs, execution is aborted; if a warning occurs, execution continues.

There is basic help information built into the interfaces. In R you can type a question mark followed by the function name, e.g. `?dstable`, without parenthesis, to get help on the function.

The STABLE library is not reentrant; only one user should be using the library at once.

The user should be aware that these routines attempt to calculate quantities related to stable distributions with high accuracy. Nevertheless, there are times when the accuracy is limited. If α is small, the pdf and cdf have very abrupt changes and are hard to calculate. When some quantity is small, e.g. the cdf of the light tail of a totally skewed stable distribution, the routines may only be accurate to approximately ten decimal places. There are certain values of the parameters (α near 2, α near 1, β near ± 1 , etc.) where there are complicated numerical problems with calculations. In these cases, the STABLE program may approximate values by rounding parameters. For example, if you try to calculate a stable pdf or cdf for $\alpha = 1.009$ and $\beta = 0.009$, the STABLE program will round to $\alpha = 1$ and $\beta = 0$, and compute the value for these values of the parameters. Likewise, when x is near 0 in the 1-parameterization, STABLE will do a linear interpolation to compute the pdf or cdf at that point. The thresholds used in rounding and linear approximation are described on page 12. You can manually reset these values, but be careful: the algorithms may yield poor values in some cases.

The remainder of this manual is a description of the functions in the STABLE library.

2.1 Basic functions

2.1.1 Test scripts

R function: `stabletest`, `mvstabletest`, `discretetest`

These will test most of the STABLE routines and can be used as a source of examples on how to use the functions.

2.1.2 Stable densities

R function: `dstable(x,alpha,beta,gamma,delta,param)`

This function computes stable density functions (pdf): $y_i = f(x_i) = f(x_i|\alpha, \beta, \gamma, \delta; \text{param})$, $i = 1, \dots, n$. The algorithm is described in Nolan (1997).

2.1.3 Stable distribution functions

R function: `pstable(x, alpha, beta, gamma, delta, param)`

This function computes stable cumulative distribution functions (cdf): $y_i = F(x_i) = F(x_i|\alpha, \beta, \gamma, \delta; \text{param})$, $i = 1, \dots, n$. The algorithm is described in Nolan (1997).

2.1.4 Stable quantiles

R function: `qstable(p, alpha, beta, gamma, delta, param)`

This function computes stable quantiles, the inverse of the cdf: $x_i = F^{-1}(p_i)$, $i = 1, \dots, n$. The quantiles are found by numerically inverting the cdf. Extreme tail quantiles may be hard to find because of subtractive cancelation and the fact that cdf calculations may only be accurate to 10 decimal places; see the notes below.

Note that the accuracy of the inversion is determined by two internal tolerances. (See Section 2.3.3.) (1) tolerance 10 is used to limit how low a quantile can be searched for. The default value is $p = 10^{-10}$: quantiles below p will be set to the left endpoint of the support of the distribution, which may be $-\infty$. Likewise, quantiles above $1 - p$ will be set to the right endpoint of the support of the distribution, which may be $+\infty$. (2) tolerance 2 is the relative error used when searching for the quantile. The search tries to get full precision, but if it can't, it will stop when the relative error is less than tolerance 2.

2.1.5 Simulate stable random variates

R function: `rstable(n, alpha, beta, gamma, delta, param)`

This function simulates n stable random variates: x_1, x_2, \dots, x_n with parameters $(\alpha, \beta, \gamma, \delta)$ in parameterization `param`. It is based on Chambers et al. (1976).

2.1.6 Stable hazard function

R function: `hstable(x, alpha, beta, gamma, delta, param)`

This function computes the hazard function for a stable distribution: $h_i = f(x_i)/(1 - F(x_i))$, $i = 1, \dots, n$.

2.1.7 Derivative of stable densities

R function: `dstable.deriv(x, alpha, beta, gamma, delta, param)`

This function computes the derivative of stable density functions: $y_i = f'(x_i) = f'(x_i|\alpha, \beta, \gamma, \delta; \text{param})$, $i = 1, \dots, n$.

2.1.8 Second derivative of stable densities

R function: `dstable.secondderiv(x, alpha, beta, gamma, delta, param)`

This function computes the second derivative of stable density functions: $y_i = f''(x_i) = f''(x_i|\alpha, \beta, \gamma, \delta; \text{param})$, $i = 1, \dots, n$.

2.1.9 Stable score/nonlinear function

R function: `stable.score(x, alpha, beta, gamma, delta, param)`

This function computes the score or nonlinear function for a stable distribution: $g(x) = -f'(x)/f(x) = -(d/dx) \ln f(x)$. The routine uses `stablepdf` to evaluate $f(x)$ and numerically evaluates the derivative

$f'(x)$. Warning: this routine will give unpredictable results when $\beta = \pm 1$. The problems occur where $f(x) = 0$ is small; in this region calculations of both $f(x)$ and $f'(x)$ are of limited accuracy and their ratio can be very unreliable.

2.2 Statistical functions

2.2.1 Estimating stable parameters

R function: `stable.fit(x, method, param)`

Estimate stable parameters from the data in x_1, \dots, x_n , using `method` as described in the following table. This routine calls one of the functions described below to do the actual estimation; see those sections for references.

method value	algorithm	notes
1	maximum likelihood	$\alpha \geq 0.2$
2	quantile	$\alpha \geq 0.1$
3	empirical characteristic function	$\alpha \geq 0.1$
4	fractional moment	$\alpha \geq 0.4, \beta = \delta = 0$, uses $p = 0.2$
5	log absolute moment	$\beta = \delta = 0$
6	modified quantile	$\alpha \geq 0.4$
7	U statistic method	$\beta = \delta = 0$

Note that the fractional moment, log absolute moment, and U statistic methods do not work when there are zeros in the data set. They also assume that the distribution is symmetric and centered at 0; if either of these assumptions are not valid, the estimators are unreliable.

2.2.2 Maximum likelihood estimation

R function: `stable.fit.mle(x, param)`

Estimate the stable parameters for the data in x_1, \dots, x_n , in parameterization `param` using maximum likelihood estimation. The likelihood is numerically evaluated and maximized using an optimization routine. This program and the numerical computation of confidence intervals below are described in Nolan (2001). For speed reasons, the quick log likelihood routine is used to approximate the likelihood; this is where the restriction $\alpha \geq 0.2$ comes from.

2.2.3 Maximum likelihood estimation with restricted parameters

R function: `stable.fit.mle.restricted(data, theta, restriction, param)`

This is a modified version of maximum likelihood estimation, where some parameters can be estimated while the others are restricted to a fixed value. The function takes an input value `theta = {alpha, beta, gamma, delta}` and if `restriction[i] = 1`, then `theta[i]` is fixed; to allow a parameter to vary, set `restriction[i] = 0`. The function then searches over the unrestricted parameters to maximize the likelihood.

2.2.4 Maximum likelihood estimation with search control

R function: `stable.fit.mle0(data, method, gamma0, delta0, ftol, tlower, tupper)`

This is maximum likelihood estimation with greater control over the search and ranges for the parameters. It is used internally and always uses the 0 parameterization.

2.2.5 Quantile based estimation

R function: `stable.fit.quantile(x, param)`

Estimate stable parameters for the data in x , using the quantile based on the method of McCulloch (1986). It sometimes has problems when α is small, e.g. $\alpha < 1/2$, and the data is highly skewed. Try the modified version below in such cases.

2.2.6 Empirical characteristic function estimation

R function: `stable.fit.ecf(x, gamma0, delta0, param)`

Estimate stable parameters for the data in x using the empirical characteristic function based method of Koutrovelis-Kogon-Williams, described in Kogon and Williams (1998). An initial estimate of the scale `gamma0` and the location `delta0` are needed to get accurate results. We recommend using the quantile based estimates of these parameters as input to this routine.

2.2.7 Fractional moment estimation

R function: `stable.fit.fractional.moment(x, p)`

Estimate stable parameters for the data in x , using the fractional moment estimator as in Nikias and Shao (1995). This routine only works in the symmetric case, it will always return $\beta = 0$ and $\delta = 0$. In this case the 0-parameterization coincides with the 1-parameterization, so there is no need to specify parameterization. `p` is the fractional moment power used. A reasonable default value is $p = 0.2$; it is required that $p < 1$. Take $p < \alpha/2$ to get reasonable results.

This method does not work if there are zeros in the data set - negative sample moments do not exist. Remove zero values (and possibly values close to 0) from the data set if you want to use this method. The method assumes the data is symmetric and centered at 0; departures from either assumption may generate unreliable estimates.

2.2.8 Log absolute moment estimation

R function: `stable.fit.logabs(x)`

Estimate stable parameters for the data in x , using the log absolute moment method as in Nikias and Shao (1995), Section 5.7 and Zolotarev (1986), Section 4.1. This routine only works in the symmetric case, it will always return $\beta = 0$ and $\delta = 0$. In this case the 0-parameterization coincides with the 1-parameterization, so there is no need to specify parameterization.

The log absolute moment method does not work when there are zeros in the data set, because $\log|x|$ is undefined when x is 0. Remove zero values (and possibly values close to 0) from the data set if you want to use this method. The method assumes the data is symmetric and centered at 0; departures from either assumption may generate unreliable estimates.

2.2.9 Quantile based estimation, version 2

R function: `stable.fit.quantile2(x, param)`

Estimate stable parameters for the data in x , using a modified quantile method of Nolan (2017). It should work for any values of the parameters, but some extreme values may be unreliable.

2.2.10 U statistic based estimation

R function: `stable.fit.u.est(x)`

Estimate stable parameters for the data in x , using the method of Fan (2006). The U statistic method does not work when there are zeros in the data set. Remove zero values (and possibly values close to 0) from the data set if you want to use this method. The method assumes the data is symmetric and centered at 0; departures from either assumption may generate unreliable estimates.

2.2.11 Confidence intervals for ML estimation

R function: `stable.mle.conf.int(theta, z, n)`

This routine finds confidence intervals for maximum likelihood estimators of all four stable parameters. The routine returns a vector `sigtheta` of half widths of the confidence interval for each parameter in `theta=(alpha,beta,gamma,delta)`. These values depend on the confidence level you are seeking, specified by z , and the size of the sample n . The z value is the standard critical value from a normal distribution, i.e. use $z = 1.96$ for a 95% confidence interval. For example, the point estimate of α is `theta[1]`, and the confidence interval is `theta[1]±sigtheta[1]`. For β , the confidence interval is `theta[2]±sigtheta[2]`, for γ , the confidence interval is `theta[3]±sigtheta[3]`, For δ , the confidence interval is `theta[4] ± sigtheta[4]`. These values do not make sense when a parameter is at the boundary of the parameter space, e.g. $\alpha = 2$ or $\beta = \pm 1$.

These values are numerically approximated using a grid of numerically computed values in Nolan (2001). The values have limited accuracy, especially when $\alpha \leq 1$.

2.2.12 Information matrix for stable parameters

R function: `stable.mle.info.matrix(theta)`

Returns the 4×4 information matrix for maximum likelihood estimation of the stable parameters for parameter values `theta`. This is done in the continuous 0-parameterization. These are approximate values, interpolated from a grid of numerically computed values in Nolan (2001) for $\alpha \geq 0.5$. The values have limited accuracy, especially when $\alpha \leq 1$.

2.2.13 Log-likelihood computation

R function: `stable.loglik(x, theta, param)`

Compute the log-likelihood of the data, assuming an underlying stable distribution with the specified parameters.

2.2.14 Chi-squared goodness-of-fit test

R function: `stable.chisq(data, theta, param, nclass)`

Compute chi-squared goodness-of-fit statistic for the data in x_1, \dots, x_n using `nclass` equally probable classes/bins. This test only looks at proportion of the data in each class, not how it is spread within that bin. This is particularly a problem with the end classes, which are infinite regions. This test does not consider the tail decay. There is also an issue with significance values when parameters are estimated from the data.

2.2.15 Kolmogorov-Smirnov goodness-of-fit test

R function: `stable.ks.gof(x, theta, method, param)`

This function computes the Kolmogorov-Smirnov two-sided test statistic:

$$D = \sup_{-\infty < x < \infty} |F(x) - \hat{F}(x)|,$$

where $F(\cdot)$ is the stable cdf with parameters $\alpha = \text{theta}[1]$, $\beta = \text{theta}[2]$, $\gamma = \text{theta}[3]$, $\delta = \text{theta}[4]$ and $\hat{F}(\cdot)$ is the sample cdf of the data in x . Use `method=0` for quick computations (the fast approximation is used to compute cdf), use `method=1` for slower computations (the slow method is used to compute cdf). The routine returns the observed value of D and an estimate of the tail probability $P(D > d)$, i.e. the significance level of the test. This tail probability is calculated using Stephen's approximation to the limiting distribution, e.g. $(n^{1/2} + 0.12 + 0.11n^{-1/2})D$ is close to the limiting Smirnov distribution. This is close to $n^{1/2}D$ for large n , and a better approximation on the tails for small n . Note, this calculation is not very accurate if the tail probability is large, but these cases aren't of much interest in a goodness-of-fit test. (If you don't like this approximation, the function returns D , and you can compute your own tail probability.) **WARNING:** the computation of the significance level is based on the assumption that the parameter values $\text{theta}=(\alpha, \beta, \gamma, \delta)$ were chosen independently of the data. If the parameters were estimated from the data, then this tail probability will be an overestimate of the significance level.

2.2.16 Likelihood ratio test

R function: `stable.lrt(x, abnd, bbnd)`

This function computes the likelihood ratio L_0/L_1 , where L_0 is the maximum likelihood of the data x under the assumption that x is an i.i.d. sample from a stable distribution with α and β restricted to the range $\text{abnd}[1] \leq \alpha \leq \text{abnd}[2]$ and $\text{bbnd}[1] \leq \beta \leq \text{bbnd}[2]$, and L_1 is the maximum likelihood of the data under an unrestricted stable model. The function computes the maximum likelihood using the quick approximation to stable likelihoods, so is limited to α in the range $[0.4, 2]$.

The vector `results` will contain the results of the computations:

```
results[1] = ratio of the likelihoods
results[2] = -2*log(ratio of likelihoods)
results[3] = log likelihood of the data for the restricted H0
results[4] = log likelihood of the data for the unrestricted H1
results[5] = estimated value of alpha under H0
results[6] = estimated value of beta under H0
results[7] = estimated value of gamma under H0
results[8] = estimated value of delta under H0
results[9] = estimated value of alpha without assuming H0
results[10] = estimated value of beta without assuming H0
results[11] = estimated value of gamma without assuming H0
results[12] = estimated value of delta without assuming H0
```

Note that under the standard assumptions, `results[2]` converges to a chi-squared distribution with d.f. = (# free parameters in H1 parameter space - # free parameters in H0 parameter space) as the sample size tends to ∞ .

For example, to compute the likelihood ratio test for the null hypothesis H0: data comes from a normal distribution vs H1: data comes from stable distribution, use `abnd=(2,2)` and `bbnd=(0,0)`, in which case `results[2]` will have 2 d.f. To test H0: data comes from a symmetric stable distribution vs H1: data comes from a general stable distribution, use `abnd=(0.4,2)` and `bbnd=(0,0)`, in which case `results[2]` will have 3 d.f.

2.2.17 Stable regression

R function: `stable.regression(X, y, trimprob, symmetric, param)`

Computes linear regression coefficients $\theta_1, \theta_2, \dots, \theta_k$ for the problem

$$y_i = \theta_1 x_{i,1} + \theta_2 x_{i,2} + \dots + \theta_k x_{i,k} + e_i, \quad i = 1, \dots, n$$

where the error term e_i has a stable distribution. The algorithm uses maximum likelihood and is described in Nolan and Ojeda-Revah (2013). In matrix form, the equation is $y = X\theta + e$.

y is a vector of length n of observed responses. X is a $n \times k$ matrix, with the columns of X representing the variables and the rows representing the different observations. NOTE: if you want an intercept term, you must include a column of ones in the X matrix. Typically one sets the first column of X to ones, and then θ_1 is the intercept.

`trimprob` is a vector of length 2, e.g. (0.1,0.9), which gives the lower and upper quantiles for the trimming. (Trimmed regression trims off extreme values and then performs ordinary least squares regression. The resulting coefficients are used to get an initial estimate of the stable regression coefficients.) The variable `symmetric` can be used to force the fitting program to assume symmetry in the error terms e_i . `param` is the parameterization used and must be 0, 1, or 2; the default is parameterization 2.

This function returns a structure with different fields.

- `theta` is the vector of regression coefficients found by maximum likelihood
- `theta_ols` is the initial vector of coefficients from the OLS regression
- `theta_trim` is the initial vector of coefficients from the trimmed regression
- `psi=(alpha,beta,gamma,delta)` are the stable parameters estimated from the residuals. They can be regarded as nuisance parameters if you only care about the regression coefficients. Note that all parameters are in the `param`-parameterization.
- `param` the parameterization used.
- `symmetric` whether regression was restricted to the symmetric case.

Note that in the non-Gaussian stable case, some of the traditional assumptions in regression are no longer true. It is not generally the case that $Ee_i = 0$, so the estimates may not be unbiased. First, if $\alpha \leq 1$, the tails of the stable distribution are very heavy and Ee_i is undefined. Second, in the non-symmetric case, i.e. $\beta \neq 0$, we do not require $Ee_i = 0$. Also, it is not generally the case that the regression line goes through the center of the data. This is not an error; it is a consequence of regression with non-symmetric residuals and how the parameterization centers the distribution.

What happens in the non-symmetric case depends on the parameterization used. Using `param=0` will give a well conditioned problem, but the regression line will not go through the center of the data. Using `param=1` will guarantee that $Ee_i = 0$ (when $\alpha > 1$ and the expectation exists), but has two consequences: the numerics are poorly conditioned if α is near 1 so the parameter estimates are very sensitive, and the line can be arbitrarily far away from the center of the data. Using `param=2` will guarantee that the mode of e_i is zero, and the regression line will go through the center of the data points.

Note: There is experimental code in R to estimate confidence intervals for the regression parameters. Please contact us if you want to try that code.

2.3 Informational/utility functions

2.3.1 Version information

R function: `stable.version()`

`stableversion()` returns a string with version information.

2.3.2 Modes of stable distributions

R function: `stable.mode(theta,param)`

Returns the mode of a $S(\alpha = \text{theta}[1], \beta = \text{theta}[2], \gamma = \text{theta}[3], \delta = \text{theta}[4]; \text{param})$ distribution. If $\beta \neq 0$, the mode is determined by a numerical search of the pdf.

2.3.3 Set internal tolerance

R function: `stable.set.tolerance(inum, value)`

Sets the value of internal tolerances that are used during computations. You change these values at your own risk: computation times can become very long, inaccuracy can accumulate, and some choices of the parameters can cause infinite loops.

inum	meaning
0	relative error for pdf numerical integration
1	relative error for cdf numerical integration
2	relative error for quantile search
3	alpha and beta rounding
4	x tolerance near zeta
5	exponential cutoff
6	peak/strim location tolerance
7	stabletrim tolerance
8	minimum alpha
9	minimum xtol
10	threshold for quantile search
11	x tolerance

2.3.4 Get internal tolerance

R function: `stable.get.tolerance(inum)`

Returns the value of the internal tolerances, see the preceding function for the meanings of each variable.

2.3.5 Convert between parameterizations

R function: `stable.convert(param, theta, newparam)`

Convert from the parameters given in `theta=(alpha, beta, gamma, delta)` given in the `param`-parameterization to the parameters `thetaneu` given in the `newparam`-parameterization. Currently `param` and `newparam` are restricted to the values 0,1,2 and 3.

2.3.6 Omega function

R function: `stable.omega(u, alpha, beta, param)`

Compute the function $\omega(u_i|\alpha, \beta; k)$, $i = 1, \dots, n$ where

$$\begin{aligned} \omega(u|\alpha, \beta; 0) &= \begin{cases} |u|^\alpha [1 + i\beta(\tan \frac{\pi\alpha}{2})(\text{sign } u)(|u|^{1-\alpha} - 1)] & \alpha \neq 1 \\ |u| [1 + i\beta\frac{2}{\pi}(\text{sign } u) \ln |u|] & \alpha = 1, \end{cases} & (3) \\ \omega(u|\alpha, \beta; 1) &= \begin{cases} |u|^\alpha [1 - i\beta(\tan \frac{\pi\alpha}{2})(\text{sign } u)] & \alpha \neq 1 \\ |u| [1 + i\beta\frac{2}{\pi}(\text{sign } u) \ln |u|] & \alpha = 1. \end{cases} \end{aligned}$$

These functions are from the characteristic functions of standardized univariate stable distributions: if $Z \sim \mathbf{S}(\alpha, \beta, 1, 0; k)$, then $E \exp(iuZ) = \exp(-\omega(u|\alpha, \beta; k))$. As before, $k = 0$ or $k = 1$ correspond to two different parameterization. The function returns two vectors containing the real and imaginary parts of $\omega(u|\alpha, \beta; k)$

2.4 Series approximations to basic distribution functions

These functions use the Bergstrom series for stable densities and cdfs, which are only defined for $\alpha \neq 1$.

2.4.1 Series approximation of stable pdf around the origin

R function: `dstable.series.origin(x, alpha, beta, gamma, delta, param, nterms)`

Computes the stable probability distribution function using a series approximation with `nterms` in it. This function is best used to calculate the density near the origin in the 1-parameterization. The series is not defined for $\alpha = 1$. Note that `nterms=1` corresponds to a constant term, `nterms=2` corresponds to a linear term, etc.

2.4.2 Series approximation of stable cdf around the origin

R function: `pstable.series.origin(x, alpha, beta, gamma, delta, param, nterms)`

Computes the stable cumulative distribution function using a series expansion with `nterms` in it. This function is best used to calculate the cdf near the origin in the 1-parameterization. The series is not defined for $\alpha = 1$. Note that `nterms=1` corresponds to a constant term, `nterms=2` corresponds to a linear term, etc.

2.4.3 Series approximation of stable pdf at the tail

R function: `dstable.series.tail(x, alpha, beta, gamma, delta, param, nterms)`

Computes the stable probability distribution function using a series approximation with `nterms` in it. This function is best used to calculate points on the tail of a distribution. The series is defined only for $x > 0$. (For $x < 0$, replace x by $-x$ and β by $-\beta$. The series is not defined for $\alpha = 1$.)

2.4.4 Series approximation of stable cdf at the tail

R function: `pstable.series.tail(x, alpha, beta, gamma, delta, param, nterms)`

Computes the stable cumulative distribution function using a series approximation with `nterms` in it. This function is best used to calculate points on the tail of a distribution. The series is defined only for $x > 0$. (For $x < 0$, replace x by $-x$ and β by $-\beta$. The series is not defined for $\alpha = 1$.)

2.5 Faster approximations to basic functions

The functions described in preceding sections are accurate, but can take a long time to compute. For evaluating a single pdf or cdf at a single set of parameter values, they are fine. However, when the functions must be evaluated many times, the previous routines are slow. For example, when estimating stable parameters by maximum likelihood estimation, the likelihood is evaluated at each data point for a large number of parameter values during the numerical search for the point where the likelihood is maximized. In these cases, speed is more desirable than great accuracy.

The functions described below are approximations to the functions above, and are based on pre-computed values using those basic functions. They are designed to evaluate the quantity of interest at many x values for fixed values of α and β . Each routine has a setup time, and if you change α or β , that setup code must be rerun. It can be slower to run these routines than the basic routines above if you only want to calculate the quantity at a few x values. These routines work for $0.2 \leq \alpha \leq 2$ and all $-1 \leq \beta \leq 1$.

2.5.1 Quick stable density computation

R function: `dstable.quick(x, alpha, beta, gamma, delta, param)`

Call is identical to Section 2.1.2, results are approximately the same.

2.5.2 Quick stable cumulative computation

R function: `pstable.quick(x, alpha, beta, gamma, delta, param)`

Call is identical to Section 2.1.3, results are approximately the same.

2.5.3 Quick stable log pdf computation

R function: `dstable.log.quick(x, alpha, gamma, delta, param)`

Approximates $\log(f(x))$ for stable distributions.

2.5.4 Quick stable quantile computation

R function: `qstable.quick(p, alpha, beta, gamma, delta, param)`

Call is identical to Section 2.1.4, but much faster. Note the comments in that section about extreme upper quantiles.

2.5.5 Quick stable hazard function computation

R function: `hstable.quick(x, alpha, beta, gamma, delta, param)`

Call is identical to Section 2.1.6.

2.5.6 Quick stable likelihood computation

R function: `stable.loglik.quick(x, theta, param)`

Call is identical to Section 2.2.13.

2.5.7 Quick stable score/nonlinear function

R function: `stable.score.quick(x, alpha, beta, gamma, delta, param, method)`

This function approximates the score or nonlinear function for a stable distribution: $g(x) = -f'(x)/f(x) = -(d/dx) \ln f(x)$. The algorithm used depends on the value of `method`. When `method=1`, `stableqkpdf` is used to compute $f(x)$ and in the numerical evaluation of $f'(x)$. When `method=2`, `stablescorefn` is used to compute $g(x)$ on a grid, then a spline is fit to those values. The resulting spline is used to approximate $g(x)$. If n is large, this is noticeably faster than either `stablescorefn` or `method=1` above. When `method=3`, a rational function approximation is used to approximate $g(x)$. This is the fastest method, but the accuracy depends on the values of α and β . If α is between 1 and 1.9 and β is near 0, the approximation is good.

2.6 Discrete stable distributions

Given a stable distribution $X \sim S(\alpha, \beta, \gamma, \delta; \text{param})$ and a pair of cutoff values $a < b$, the random variable

$$Y = \text{integer part of } \max(a, \min(X, b))$$

is a discrete stable distribution. These distributions arise in signal processing where a continuous quantity is quantized/digitized and limited accuracy is kept. It is assumed that the cutoff values are integers. The saturation probability is $P(X < a - 1/2) + P(X > b + 1/2)$, and is a measure of how much of the distribution is lost by truncating at the cutoff values. In the routines below, the cutoff is specified by a vector of length 2: `cutoff=(a, b)`. In this section X will always refer to the continuous stable distribution, while Y will always refer to a discrete/quantized/integer valued distribution.

In the internal routines, the x values are integers. The matlab/R/Mathematica interfaces use double precision values.

2.6.1 Discrete stable density

R function: `ddiscstable(x, alpha, beta, gamma, delta, cutoff, param)`

Calculates $f_i = P(Y = x_i)$, $i = 1, \dots, n$.

2.6.2 Quick discrete stable density

R function: `ddiscstable.quick(x, alpha, beta, gamma, delta, cutoff, param)`

Calculates $f_i = P(Y = x_i)$, $i = 1, \dots, n$. Faster than above, less accurate.

2.6.3 Discrete stable cumulative distribution function

R function: `pdiscstable(x, alpha, beta, gamma, delta, cutoff, param)`

Calculates $F_i = P(Y \leq x_i)$, $i = 1, \dots, n$.

2.6.4 Quick discrete stable cumulative distribution function

R function: `pdiscstable.quick(x, alpha, beta, gamma, delta, cutoff, param)`

Calculates $F_i = P(Y \leq x_i)$, $i = 1, \dots, n$. Faster than above, less accurate.

2.6.5 Simulate discrete stable random variates

R function: `rdiscstable(n, alpha, beta, gamma, delta, cutoff, param)`

Simulates discrete stable random variates with the specified parameters and cutoffs.

2.6.6 Simulate discrete stable random variates with specified saturation probability

R function: `rdiscstable2(n, alpha, beta, delta, psaturation, cutoff, param)`

Simulates discrete stable random variates, where the scale is computed internally to make the saturation probability=`psaturation`. Note that in cases where the stable parameters are passed individually, `gamma` is NOT used. In the cases where the vector `theta` is used, the value of $\gamma = \text{theta}[3]$ is ignored. The following function is used to compute γ , then the previous function is called to generate the values.

2.6.7 Find scale γ to have a specified saturation probability for a discrete stable distribution

R function: `discstable.find.gamma(alpha, beta, delta, saturation, cutoff, param)`

Given α , β , δ and `cutoff=(a,b)`, the scale γ is computed to get the requested saturation probability, e.g. `psaturation=P(X < a - 1/2) + P(X > b + 1/2)`.

2.6.8 Discrete maximum likelihood estimation

R function: `stable.fit.dmle(x, method, cutoff, param)`

Estimate the stable parameters for the discrete stable data in x_1, \dots, x_n , in parameterization `param` using maximum likelihood estimation. The likelihood is numerically evaluated and maximized using an

optimization routine. When `method=1`, `stablepdfdiscrete` is used to calculate likelihood, when `method=2`, symmetry is assumed ($\beta = 0$) and a faster method is used to compute the likelihood.

3 Multivariate Stable Introduction

To specify a multivariate stable distribution $\mathbf{X} = (X_1, X_2, \dots, X_d)^T$ in d dimensions requires an index of stability $\alpha \in (0, 2]$, a finite Borel measure Λ on the unit sphere $\mathbb{S} = \{\mathbf{s} \in \mathbb{R}^d : |\mathbf{s}| = 1\}$ and a shift vector $\boldsymbol{\delta} \in \mathbb{R}^d$. The measure Λ is called the spectral measure of the distribution. The joint characteristic function of $\mathbf{X} \sim \mathbf{S}(\alpha, \Lambda, \boldsymbol{\delta}; k)$ is given by:

$$E \exp(i \langle \mathbf{u}, \mathbf{X} \rangle) = \exp \left(- \int_{\mathbb{S}} \omega_k(\langle \mathbf{u}, \mathbf{s} \rangle | \alpha, 1; k) \Lambda(ds) + i \langle \mathbf{u}, \boldsymbol{\delta} \rangle \right),$$

where $\omega(u|\alpha, \beta; k)$ is defined in (3). As in one dimension, the 1-parameterization is more common in theoretical research, while the 0-parameterization is better suited to computation and statistical problems. Here and below, $\langle \mathbf{u}, \mathbf{X} \rangle = \mathbf{u} \mathbf{X}^T = u_1 X_1 + \dots + u_d X_d$ is the inner product. Symmetric stable distributions are defined by the condition $\mathbf{X} \stackrel{d}{=} -\mathbf{X}$, which is equivalent to Λ being a symmetric measure on \mathbb{S} , i.e. $\Lambda(A) = \Lambda(-A)$ for any Borel subset $A \subset \mathbb{S}$. As in the univariate case, in the symmetric case the 0-parameterization and the 1-parameterization coincide.

The general case is beyond current computational capabilities, but several special cases: isotropic (radially symmetric), elliptical, independent components and discrete spectral measure are computationally accessible.

isotropic The spectral measure is continuous and uniform, leading to isotropic/radial symmetry for the distribution. The characteristic function is

$$E \exp(i \langle \mathbf{u}, \mathbf{X} \rangle) = \exp(-\gamma_0^\alpha |\mathbf{u}|^\alpha + i \langle \mathbf{u}, \boldsymbol{\delta} \rangle). \quad (4)$$

elliptical The characteristic function is

$$E \exp(i \langle \mathbf{u}, \mathbf{X} \rangle) = \exp \left(-(\mathbf{u}^T R \mathbf{u})^{(\alpha/2)} + i \langle \mathbf{u}, \boldsymbol{\delta} \rangle \right) \quad (5)$$

where R is a positive definite matrix. ($R = \gamma_0^2 I$ is equivalent to the isotropic case above.) More information on this accessible class of distributions is given in Nolan (2010).

independent components If components are independent with $X_j \sim \mathbf{S}(\alpha, \beta_j, \gamma_j, \delta_j; k)$, then the characteristic function is

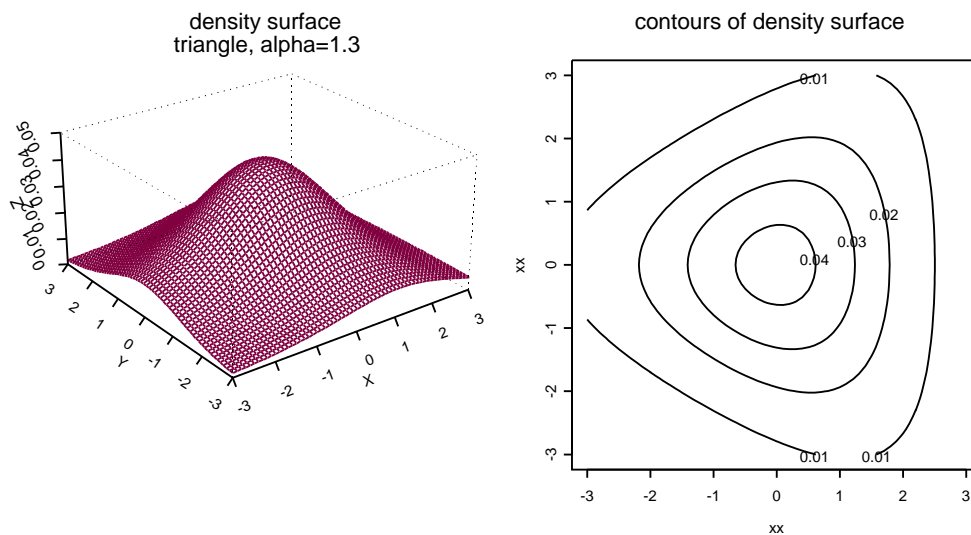
$$E \exp(i \langle \mathbf{u}, \mathbf{X} \rangle) = \exp \left(- \sum_{j=1}^d \omega(u_j | \alpha, \beta_j; k) \gamma_j^\alpha + i \langle \mathbf{u}, \boldsymbol{\delta} \rangle \right) \quad (6)$$

This is a special case of the discrete spectral measure below: the spectral mass is concentrated on the points where the coordinates axes intersect the unit sphere.

discrete When the spectral measure is discrete with mass λ_j at $\mathbf{s}_j \in \mathbb{S}$, $j = 1, \dots, m$ the characteristic function is

$$E \exp(i \langle \mathbf{u}, \mathbf{X} \rangle) = \exp \left(- \sum_{j=1}^m \omega(\langle \mathbf{u}, \mathbf{s}_j \rangle | \alpha, 1; k) \lambda_j + i \langle \mathbf{u}, \boldsymbol{\delta} \rangle \right) \quad (7)$$

This discrete class is dense in the class of all stable distributions: any finite spectral measure Λ can be approximated by a discrete measure, see Byczkowski et al. (1993). Below is a plot of the density surface of a bivariate stable density with three point masses, each of weight 1 at locations $(\cos(\pi/3), \sin(\pi/3))$, $(-1, 0)$, and $(\cos(5\pi/3), \sin(5\pi/3))$. (This plot was produced using the test script `mvstabletest`, see Section 2.1.)



4 Multivariate Stable Functions

Since the specification of a multivariate stable distribution is somewhat cumbersome, a different approach from the univariate case is taken in these routines. Two steps are needed to work with a multivariate stable distribution. First, the distribution is specified by calling a function to define the distribution. Second, call a separate functions to compute densities, cumulatives, simulate, etc.

The programs for working with multivariate stable distributions are less well developed and generally limited to 2 dimensions. At the current time, when dimension $2 < d \leq 100$, you can:

- simulate using `mvstablern`
- fit multivariate data with an elliptical model
- calculate the pdf using `mvstablepdf` if the components are independent OR the spectral measure has exactly d point masses OR the distribution is isotropic or elliptical
- calculate the cdf using `mvstablecdf` if the components are independent
- calculate the cdf using `mvstablecdfMC` by Monte Carlo estimation for any type of distribution

The accuracy of the multivariate pdf and cdf calculations are limited. In all cases, X is a column vector, this is important to remember when you specify x for calculating the multivariate pdf or cdf.

4.1 Define multivariate stable distribution

STABLE has the ability to work with multiple distributions. When a multivariate stable distribution is defined, a ‘distribution descriptor’ is returned. That descriptor must be used when computing quantities for that distribution. Note: The descriptor should not be changed by a user. The descriptor may change between calls, and contents may vary in future versions of STABLE.

There are different functions used to define each of the different types of distributions that STABLE can work with. They are described below.

A simple R example that defines two distributions and works with them is:

```
# Define three bivariate stable distributions: one isotropic,
# one with indep. components. one with elliptical contours
dist1 <- mvstable.isotropic(alpha=1.3,d=2,gamma0=1,delta=c(0,0))
```

```

dist2 <- mvstable.indep(alpha=1.3,beta=c(0,0),gamma=c(1,1),delta=c(0,0),0)
dist3 <- mvstable.elliptical( alpha=1.3, R=matrix(c(1,.5,.5,1)) )
# compute the pdf for each distributions:
x <- matrix( c(0,0,0,1,1,0,1,1), 2, 4)
y1 <- dmvstable(dist1,x)
y2 <- dmvstable(dist2,x)
y3 <- dmvstable(dist3,x)
# simulate from each distributions
z1 <- rmvstable(dist1,10000)
z2 <- rmvstable(dist2,10000)
z3 <- rmvstable(dist3,10000)

```

4.1.1 Independent components

R function: `mvstable.indep(alpha, beta, gamma, delta, param)`

Define a multivariate stable distribution with dimension $d \leq 100$ and independent components with characteristic function (6). `beta`, `gamma` and `delta` should be vectors of length $d =$ the dimension of the distribution.

4.1.2 Isotropic stable

R function: `mvstable.isotropic(alpha, d, gamma0, delta)`

Define a multivariate isotropic stable distribution with with dimension $d \leq 100$ and characteristic function (4). d is the dimension of the distribution, `gamma0` is the scale parameter, `delta` is the location vector.

4.1.3 Elliptical stable

R function: `mvstable.elliptical(alpha, R, delta)`

Define a multivariate elliptically contoured/sub-Gaussian stable distribution with dimension $d \leq 100$ and with characteristic function (5). The dimension of the distribution is determined from the size of `R`, a positive definite $d \times d$ shape matrix, and `delta` is the location vector.

4.1.4 Discrete spectral measure

R function: `mvstable.discrete.spec.meas(alpha, s, lambda, delta, beta, param)`

Define a multivariate stable distribution with discrete spectral measure with dimension $d \leq 100$ and characteristic function (7). `s` should be a $d \times nlambda$ matrix specifying the location of the point masses as columns, `lambda` should be a row vector of length `nlambda` containing the weights. `beta` should be a row vector of length `nlambda` specifying the skewness at each point mass. `delta` is the shift as a column vector. `param` is the parameterization, must be 0 or 1.

The spectral measure is defined by putting mass $lambda[j]*(1+beta[j])/2$ at s_j and mass $lambda[j]*(1-beta[j])/2$ at $-s_j$. Setting all `beta` equal to 1 gives the standard definition of a spectral measure, with mass $lambda[j]$ at $s[j]$. Setting all `beta` equal to 0 guarantees that the distribution is symmetric, putting weight $lambda[j]/2$ at $\pm s_j$. If any element of `beta` is not 0, the distribution is assumed to be nonsymmetric. (It is possible to manually make a spectral measure symmetric with nonzero `beta` by defining antipodal points and weights and values of `beta` that balance correctly. However, STABLE does not detect this.) Some parts of STABLE are significantly faster and more accurate in the symmetric case, e.g. density calculations and simulations.

4.1.5 Discrete spectral measure in 2 dimensions

R function: `mvstable.discrete.spec.meas2d(alpha, angle, lambda, beta, delta, param, degrees=FALSE)`

Define a bivariate stable distribution with discrete spectral measure. This is a special case of the previous function. In two dimensions the locations of the point masses can be specified by angles: `angle[j]` gives the angle (in radians) of the location of $s_j = (\cos(\text{angle}[j]), \sin(\text{angle}[j]))$. In R, the optional argument `degrees` can be set `TRUE` to use degrees instead of radians.

There are several special cases that are handled differently internally:

- When `angle` and `lambda` are of length 2, densities can be calculated in terms of univariate densities.
- The special case of the previous one is when `angle=(0, $\pi/2$)`. This corresponds to a distribution with independent components. Both density and cdf are calculated in terms of products of univariate density and cdf respectively.
- If all elements of `beta` are 0, the distribution is symmetric. Cumulative distribution function calculations only work in the symmetric case (though Monte Carlo based cdf estimation works for any case you can simulate, including skewed.)

4.1.6 Undefine a stable distribution

R function: `mvstable.undefine(dist)`

Clears the definition of the stable distribution `dist`. When a multivariate stable distribution is defined by one of the above functions, the STABLE library allocates memory for a distribution descriptor. The amount of memory used is not large, and this should not normally be a problem. But if your program loops and defines many multivariate stable distributions, you could have a memory problem as the descriptors accumulate. Calling this function when you are done with a stable distribution will deallocate the memory.

A technical detail: distribution descriptors are allocated internally in the STABLE library, not in the workspace of the user session. If you exit, these distribution descriptors are lost, EVEN IF YOUR SESSION INFORMATION IS SAVED. You must redefine all multivariate distributions when you restart the STABLE library.

4.2 Basic functions

4.2.1 Density function

R function: `dmvstable(dist, x)`

Computes the density $f(\mathbf{x})$ for stable distribution `dist` at each value in `x`. Note: this routine assumes that the density exists. The density will not exist if the spectral measure is supported on a proper linear subspace of the domain.

When dimension $d > 2$, the pdf can be calculated in special cases:

- the components are independent components
- the spectral measure is discrete AND the number of point masses is equal the dimension of the problem
- the isotropic case
- the elliptical case

Otherwise, only 2-dimensional computations can be done. The symmetric case uses the method in Abdul-Hamid and Nolan (1998), the nonsymmetric case uses the method in Nolan and Rajput (1995). The symmetric case is faster and more accurate than the nonsymmetric case. Both routines are accurate near the center of the distribution, and have limited accuracy near the tails.

4.2.2 Cumulative function

R function: `pmvstable(dist, a, b)`

This function approximates $P(\mathbf{a} \leq \mathbf{X} \leq \mathbf{b})$. If the components are independent, this works in dimensions up to 100.

In the symmetric two-dimensional case, the probability is evaluated by numerically integrating the (numerically computed) 2-dimensional density $f(\mathbf{x})$. Due to the limited precision in the numerical calculation of the density, and the approximate nature of the integration of this density, this routine gives only a few digits of accuracy. To find the probability of an unbounded regions, it is best to truncate the region using the routine in Section 4.2.5 to find a bounded rectangle containing most of the probability.

Use the function in Section 4.2.3 to approximate in 2-dimensional nonsymmetric case or in higher dimensions.

4.2.3 Cumulative function (Monte Carlo)

R function: `pmvstable.MC(dist, a, b, n)`

This function approximates $P(\mathbf{a} \leq \mathbf{X} \leq \mathbf{b})$ by simulating n independent random vectors with the same distribution as \mathbf{X} and counting how many are in the interval $[\mathbf{a}, \mathbf{b}]$. It works for any distribution and dimension that can be simulated.

4.2.4 Multivariate simulation

R function: `rmvstable(dist, n)`

Simulate n stable random vectors from the stable distribution `dist`. This works for any distribution that can be defined in dimensions $d \geq 2$.

4.2.5 Find a 2-dimensional rectangle with probability at least p

R function: `mvstable.find.rectangle(dist, p)`

Find a number r so that the 2-dimensional rectangle $A = A(r) = [-r, r] \times [-r, r]$ has $P(\mathbf{X} \in A) \geq p$, where \mathbf{X} is a bivariate stable distribution defined by `dist`. This is used for technical calculations, e.g. in approximating the probability of unbounded regions. The method uses univariate projections and will generally give an overestimate of r . The method is less accurate for small p or if the distribution is not centered or highly skewed, it gets more accurate if p is close to 1 and the distribution is centered and symmetric.

If p is not too close to 1, one can get a better value of r by making repeated calls to the multivariate cdf function with rectangles of the form $A(r)$ and search for a value of r that makes $P(\mathbf{X} \in A(r))$ close to p . That procedure involves bivariate numerical integration will take much longer than this function.

4.3 Statistical functions

4.3.1 Estimate a discrete spectral measure - fit a stable distribution to bivariate data

R function: `mvstable.fit(x, nspectral, method1d, method2d, param)`

`x` contains the data values, `nspectral` is the number of points in the estimated spectral measure (must be divisible by 4), `method1d` is the method to use for estimating univariate stable parameters internally (see Section 2.2.1 for codes; only used if `method2d=1`), `method2d` is the method to use in estimating bivariate distribution. Use `method2d=1` for Rachev-Xin-Cheng method, `method2d=2` for projection method, `method2d=3` for empirical characteristic function method. The methods are described in Nolan et al. (2001), see Nolan and Panorska (1997) for some discussion of suggested values and diagnostics. Suggest using `nspectral=40`, `method1d=3`, `method2d=2`, `param=1`.

The function returns a list/structure that contains information about the fit, which is always done as a discrete spectral measure. The fields in the fit are: the estimated value of α , the estimated shift/location vector δ , `angle` which is a uniform grid from 0 to 2π of length `nspectral`, and `lambda` for the estimated weights at each position.

4.3.2 Estimate parameter functions

R function: `mvstable.fit.parfn2d(x, angle, methodld, param)`

Estimate the parameter functions for the bivariate data in `x`. The data is projected in each direction given by `angle` and the parameters are estimated in the `param` parameterization. `methodld` is the univariate method used to estimate the parameters (see Section 2.2.1 for codes).

The result is a matrix of dimension `length(x) × 5`. The columns of the result are (1) for the angle, (2) for the estimate of α , (3) for the estimate of β , (4) for the estimate of γ , (5) for the estimate of δ at that angle.

4.3.3 Fit an elliptical stable distribution to multivariate data

R function: `mvstable.fit.elliptical(x, methodld)`

`x` contains the data values (d -vectors), `methodld` is the method to use for estimating univariate stable parameters internally (see Section 2.2.1 for codes).

The function returns a list/structure that contains information about the fit. The fields in the fit are: the estimated value of α , the estimated shift/location vector δ , and \mathbb{R} for the estimated shape matrix. This information can be used to define a multivariate stable distribution for simulation.

4.4 Amplitude distribution

For d -dimensional random vector \mathbf{X} , the univariate quantity $R = |\mathbf{X}|$ is called the amplitude of \mathbf{X} . When \mathbf{X} is isotropic, the radial symmetry allows one to reduce the dimension of the problem to a univariate problem. The following routines compute the cdf, pdf, quantiles, simulate and estimate for amplitudes of isotropic stable random vectors. Since these are univariate quantities, and it is required that the distribution is isotropic, one does NOT have to define the isotropic distribution separately. Because of computational difficulties, these routines are limited to dimension $d \leq 100$.

4.4.1 Amplitude cumulative distribution function

R function: `pamplitude(r, alpha, gamma0, d)`

Compute the cdf of the amplitude distribution: $F_R(r) = P(R \leq r)$ for $R = |\mathbf{X}|$, where \mathbf{X} is an d -dimensional isotropic stable random vector with characteristic function $E \exp(i \langle \mathbf{u}, \mathbf{X} \rangle) = \exp(-\gamma_0^\alpha |u|^\alpha)$. Current implementation works for $\alpha \in [0.8, 2]$. There seems to be a relative error of approximately 3% for large r .

4.4.2 Amplitude density

R function: `damplitude(r, alpha, gamma0, d)`

Compute the density $f_R(r)$ where R is described above. Current implementation works for $\alpha \in [0.8, 2]$. There seems to be a relative error of approximately 3% for large r .

4.4.3 Amplitude quantiles

R function: `qamplitude(p, alpha, gamma0, d)`

Compute the quantiles of the amplitude R described above. Current implementation works for $\alpha \in [0.8, 2]$.

4.4.4 Simulate amplitude distribution

R function: `ramplitude(nr, alpha, gamma0, d)`

Simulate n i.i.d. values of the amplitude distribution R as described above. Current implementation works for $\alpha \in (0.2, 2]$.

4.4.5 Fit amplitude data

R function: `mvstable.fit.amplitude(r, d, method)`

Estimate the parameters α and γ_0 for amplitude data. `r` contains the (univariate) amplitude data values, `d` is the dimension of the underlying distribution that the amplitude data comes from, `method` is the method to use for estimating. This initial implementation allows only `method=5`, which uses method of moments on the log of the amplitude data. Other methods are planned for the future. The function returns the estimated value of α and γ_0 .

4.4.6 Amplitude nonlinear function

R function: `amplitude.nonlinfn(r, alpha, gamma0, d)`

Compute the nonlinear function (score function for the location) of the amplitude: $g(r) = -f'(r)/f(r)$, where $f(r) = f_R(r|\alpha, \gamma_0, d)$ is the amplitude pdf defined above. Current implementation works for $\alpha \in [0.8, 2]$, $d \leq 98$. There seems to be a relative error of approximately 3% for large r .

4.5 Faster approximations to multivariate routines

There are a limited number of functions for quickly calculating multivariate functions in the 2-dimensional isotropic case. Such a distribution is specified by the index of stability α , the scale γ_0 , and the location $\delta = (\delta_1, \delta_2)$. Because the description is simple, these functions use those arguments directly and do not use a distribution descriptor.

4.5.1 Quick log-likelihood for bivariate isotropic case

R function: `mvstable.loglik.isotropic2d.quick(x, alpha, gamma0, delta)`

Compute the log likelihood of the bivariate isotropic stable data in `x` with stable index `alpha`, scale `gamma0`, and location vector `delta`. An internal approximation is used to compute the single value

$$\ell(\alpha, \gamma_0, \delta | \mathbf{x}_1, \dots, \mathbf{x}_n) = \log \prod_{i=1}^n f_{\mathbf{X}}(\mathbf{x}_i | \alpha, \gamma_0, \delta).$$

This function is designed to compute the log likelihood for a fixed α many times. In this case, it is much faster than trying to compute the right hand side above using the bivariate pdf routine in Section 4.2.1. It is also more accurate than that routine, especially on the tails. The program initially computes an approximation that depends on α ; if α changes, the approximation must be recomputed and it will be slower.

4.5.2 Quick amplitude density in bivariate case

R function: `dampitude2d.quick(r, alpha, gamma0)`

Compute the amplitude function $f_R(r|\alpha, \gamma_0, d = 2)$ for a 2-dimensional isotropic stable vector `r`. For large n , it is much faster than the function in Section 4.4.2.

4.6 Bivariate discrete stable distribution

A bivariate discrete stable distribution is defined by digitizing and truncating a continuous bivariate stable distribution $\mathbf{X} = (X_1, X_2)^T$: discrete $\mathbf{Y} = (Y_1, Y_2)^T$ has components

$$Y_i = \text{integer part of } \max(a, \min(X_i, b)),$$

where `cutoff=(a, b)` are the upper and lower cutoff values. Note that the same cutoff is used for both components of \mathbf{X} . These distributions arise in signal processing where a bivariate continuous quantity is quantized/digitized and limited accuracy is kept. It is assumed that the cutoff values are integers. The saturation probability is $p_{sat} = P(X_1 < a - 1/2) + P(X_1 > b + 1/2) + P(X_2 < a - 1/2) + P(X_2 > b + 1/2)$, and is a measure of how much of the distribution is lost by truncating at the cutoff values.

In the internal routines, the x values are integers. The R, Mathematica and matlab interfaces store these integer values in double precision numbers.

4.6.1 Discrete bivariate density

R function: `ddiscstable2d(dist, x, cutoff, eps, method)`

Compute the pdf of a discrete bivariate stable distribution. `x` should be a $2 \times n$ matrix of integer values, `cutoff` is a vector of length 2 with upper and lower cutoff values for the truncation. The typical value for `cutoff` is (-128,127); both components of (X_1, X_2) are truncated at the same value. The function returns a vector `p` of length n with

$$p_i = P(\mathbf{Y} = \mathbf{x}_i) = P(Y_1 = x_{1i}, Y_2 = x_{2i}).$$

Note that `eps` is the attempted accuracy for each probability p_i , not for the total error. The probabilities are computed using the bivariate cdf function above and thus only works for symmetric stable two dimensional distributions. It's accuracy is limited: it is likely that when all possible values of `x` are used, $\sum_i p_i$ will be slightly different from 1. The current implementation is slow. The `method` variable is unused at the current time; it will be used for faster approximations in future implementations.

4.7 Multivariate informational/utility functions

4.7.1 Information about a distribution

R function: `mvstable.info(dist)`

Returns information about distribution `dist`. Useful for checking that definition.

4.7.2 Compute projection parameter functions

R function: `mvstable.parfn2d(dist, angle)`

Compute the exact parameter functions for a bivariate stable distribution. For direction $\mathbf{t} \in \mathbb{R}^2$, $(\mathbf{X}, \mathbf{t}_j)$ is univariate stable with parameters $(\alpha, \beta(\mathbf{t}), \gamma(\mathbf{t}), \delta(\mathbf{t}))$. This function computes the parameter functions $\beta(\cdot)$, $\gamma(\cdot)$ and $\delta(\cdot)$ at the values $\mathbf{t} = (\cos \text{angle}[j], \sin \text{angle}[j])$. Angles in `angle` are given in radians.

4.7.3 Multivariate convert parameterization

R function: `mvstable.convert(dist, newparam)`

Converts between multivariate stable parameterizations. `newparam` must be 0 or 1. In the interfaced versions of STABLE, the input distribution `dist` is converted to the new parameterization, and a new distribution descriptor is returned.

5 Error/return codes

An error is unrecoverable and stops execution. For example, if you ask to compute the density of a stable parameter with $\alpha = 3$, you will get a return code of 1 and your function will stop. In contrast, a warning is informational and is usually not serious. It alerts you to the fact that the results of a calculation may have some inaccuracy. For example, stable densities have radical changes of the tail behavior when $\alpha = 2$ or $\beta = \pm 1$, and the computations have small inaccuracies in them. In practical terms this usually means little, as the difference between an $\alpha = 1.99$ stable distribution and an $\alpha = 2$ stable distribution in an statistical problem is likely to be unobservable in practice.

In the S-Plus/R version of STABLE, error messages are controlled by the variable `err.level` in the command `load.stable: err.level=0` means no messages are printed, `err.level=1` means no only error messages are printed, `err.level=2` (the default) means both error and warning messages are printed.

Return codes for STABLE program are given in the tables below. Univariate routines return error codes in the range 1-99, multivariate routines return error codes in the range 100-199.

code	type	meaning
0		No error
1	error	Invalid input parameter
2	error	alpha parameter outside of tabulated values in QKSTABLE
3	error	Too many data points for internal array
4	error	Error computing the likelihood, e.g. pdf=0
5	warning	Possible approx. error while using QKSTABLE for alpha or beta near boundary
6	warning	Possible error in confidence intervals because parameter is near boundary
7	warning	alpha and/or beta rounded to a special value, adjust tol(4)
8	warning	alpha is at lower bound for search, may not have found best value for alpha
9	error	Too many bins (distinct possible values) in sdiscreteMLE
10	error	beta must be 0 to use this function
11	error	beta near +1 or -1 does not work in this function
12	error	sinc error in sfitfracmoment
13	error	Internal error in sfitlogabs
14	error	Data value near zero in sfitfracmoment or sfitlogabs
15	error	Error in subroutine
16	error	Internal error while computing derivatives
17	error	f(a) and f(b) have the same signs
18	error	Too many function evaluations
19	error	Not enough memory
20	error	X zero value
21	error	Internal error in quickstable
22	error	More Uniform(0,1) values required for simulation
23	error	2-parameterization required

Table 1: Univariate error codes

Warning code 7 can arise in several ways. The purpose of this warning is to avoid numerical problems in internal calculations that can occur near the boundary in the parameter space or to use special cases to increase speed, but to let the user know that something nonstandard is being done. In the following discussion, let $\epsilon =$ the value of tolerance(4). The default value is $\epsilon = 0.01$. (You can change the value of tolerance(4) by using the function `stablesettolerance` above, and query it's value by using function `stablegettolerance`. The default value was picked in an ad hoc way; you can make it smaller, even 0, if you wish to calculate certain quantities in one of the cases below. But be aware that numerical errors may arise.) Special cases where warning code 7 occur are:

1. α near 2: if $\alpha \in (2 - \epsilon, 2)$, then α is set to 2 and β is set to 0.
2. Near $\alpha = 1$ but not Cauchy: if $|\alpha - 1| < \epsilon$ and $|\beta| \geq \epsilon$, then α is set to 1 and β is left unchanged. This is to avoid computations involving $\beta \tan(\pi\alpha/2)$, which blows up as $\alpha \rightarrow 1$ if $\beta \neq 0$.
3. Near Cauchy case: if $|\alpha - 1| < \epsilon$ and $|\beta| < \epsilon$, then α is set to 1 and β is set to 0.

4. Near Lévy case: if $|\alpha - 1/2| < \epsilon$ and $|\beta - 1| < \epsilon$, then α is set to $1/2$ and β is set to 1 ; if $|\alpha - 1/2| < \epsilon$ and $|\beta + 1| < \epsilon$, then α is set to $1/2$ and β is set to -1 .

code	type	meaning
101	error	Invalid input parameter
102	warning	Accuracy warning, alpha < 1
103	warning	vmax exceeded in mvstablepdf
104	error	Too many points in spectral measure
105	error	nspectral must be divisible by 4
106	error	This parameterization is not allowed in this function
108	error	Distribution not defined
109	error	mvstablecdf not implemented for nonsymmetric case
110	error	Matrix is not positive definite
111	error	alpha must be at least 0.8
112	error	Definition error
113	error	Dimension is greater than the max allowed
115	error	Spline error
150	error	Not enough memory
151	error	Error in a subroutine

Table 2: Multivariate error codes

References

- Abdul-Hamid, H. and J. P. Nolan (1998). Multivariate stable densities as functions of one dimensional projections. *J. Multivar. Anal.* 67, 80–89.
- Byczkowski, T., J. P. Nolan, and B. Rajput (1993). Approximation of multidimensional stable densities. *J. Multivar. Anal.* 46, 13–31.
- Chambers, J., C. Mallows, and B. Stuck (1976). A method for simulating stable random variables. *Journal of the American Statistical Association* 71(354), 340–344.
- Fan, Z. (2006). Parameter estimation of stable distributions. *Communications in Statistics. Theory and Methods* 35, 245–256.
- Kogon, S. and D. Williams (1998). *Characteristic function based estimation of stable parameters*. In R. Adler, R. Feldman, and M. Taqqu (Eds.), *A Practical Guide to Heavy Tailed Data* pp. 311-338. Boston, MA: Birkhäuser.
- McCulloch, J. H. (1986). Simple consistent estimators of stable distribution parameters. *Communications in Statistics. Simulation and Computation* 15, 1109–1136.
- Nikias, C. L. and M. Shao (1995). *Signal Processing with Alpha-Stable Distributions and Applications*. New York: Wiley.
- Nolan, J. P. (1997). Numerical calculation of stable densities and distribution functions. *Commun. Statist. -Stochastic Models* 13, 759–774.
- Nolan, J. P. (2001). Maximum likelihood estimation of stable parameters. In O. E. Barndorff-Nielsen, T. Mikosch, and S. I. Resnick (Eds.), *Lévy Processes: Theory and Applications*. Boston: Birkhäuser.
- Nolan, J. P. (2010). Multivariate elliptically contoured stable distributions: theory and estimation. Submitted.
- Nolan, J. P. (2017). *Stable Distributions - Models for Heavy Tailed Data*. Boston: Birkhäuser. In progress, Chapter 1 online at academic2.american.edu/~jpnolan.
- Nolan, J. P. and D. Ojeda-Revah (2013). Linear and nonlinear regression with stable errors. *J. of Econometrics* 172, 186–194.
- Nolan, J. P., A. Panorska, and J. H. McCulloch (2001). Estimation of stable spectral measures. *Mathematical and Computer Modelling* 34, 1113–1122.
- Nolan, J. P. and A. K. Panorska (1997). Data analysis for heavy tailed multivariate samples. *Comm. in Stat. - Stochastic Models* 13, 687–702.
- Nolan, J. P. and B. Rajput (1995). Calculation of multidimensional stable densities. *Commun. Statist. - Simula.* 24, 551–556.
- Zolotarev, V. M. (1986). *One-dimensional Stable Distributions*, Volume 65 of Translations of mathematical monographs. American Mathematical Society. Translation from the original 1983 Russian edition.

Index

R functions

- amplitude.nonlinfn, 23
- daplitude, 22
- daplitude2d.quick, 23
- ddiscstable, 15
- ddiscstable.quick, 15
- ddiscstable2d, 24
- discstable.find.gamma, 15
- dmvstable, 20
- dstable, 5
- dstable.deriv, 6
- dstable.log.quick, 14
- dstable.quick, 13
- dstable.secondderiv, 6
- dstable.series.origin, 13
- dstable.series.tail, 13
- hstable, 6
- hstable.quick, 14
- mvstable.convert, 24
- mvstable.discrete.spec.meas, 19
- mvstable.discrete.spec.meas2d, 20
- mvstable.elliptical, 19
- mvstable.find.rectangle, 21
- mvstable.fit, 21
- mvstable.fit.amplitude, 23
- mvstable.fit.elliptical, 22
- mvstable.fit.parfn2d, 22
- mvstable.indep, 19
- mvstable.info, 24
- mvstable.isotropic, 19
- mvstable.loglik.isotropic2d.quick, 23
- mvstable.parfn2d, 24
- mvstable.undefine, 20
- pamplitude, 22
- pdiscstable, 15
- pdiscstable.quick, 15
- pmvstable, 21
- pmvstable.MC, 21
- pstable, 6
- pstable.quick, 14
- pstable.series.origin, 13
- pstable.series.tail, 13
- qamplitude, 22
- qstable, 6
- qstable.quick, 14
- ramplitude, 23
- rdiscstable, 15
- rdiscstable2, 15
- rmvstable, 21
- rstable, 6
- stable.chisq, 9
- stable.convert, 12
- stable.fit, 7
- stable.fit.dmle, 15
- stable.fit.ecf, 8
- stable.fit.fractional.moment, 8
- stable.fit.logabs, 8
- stable.fit.mle, 7
- stable.fit.mle.restricted, 7
- stable.fit.mle0, 7
- stable.fit.quantile, 8
- stable.fit.quantile2, 8
- stable.fit.u.est, 8
- stable.get.tolerance, 12
- stable.ks.gof, 9
- stable.loglik, 9
- stable.loglik.quick, 14
- stable.lrt, 10
- stable.mle.conf.int, 9
- stable.mle.info.matrix, 9
- stable.mode, 11
- stable.omega, 12
- stable.regression, 10
- stable.score, 6
- stable.score.quick, 14
- stable.set.tolerance, 12
- stable.version, 11
- stabletest, mvstabletest, discretetest, 5