PERGAMON Mathematical and Computer Modelling 34 (2001) 1073-1093

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.com/locate/mcm

Classification Rules for Stable Distributions

A. SENGUPTA
Applied Statistics Division
Indian Statistical Institute
Calcutta, India

S. ROY
Department of Statistics
University of Cork
Cork, Republic of Ireland

Abstract—Often one is interested in identifying whether the financial market for a commodity has entered a "bullish" or a "bearish" characteristic. Stable or, more commonly, α-stable distributions, have been enhanced as a popular model for stock prices, and such change in characteristics may be related to the parameter(s) of these underlying distributions. This paper deals with discrimination between two stable distributions, or in turn, with classification of a new observation into one of two stable distributions. In case the parameters are unknown, we need training samples, one each from the two populations, which are utilized to provide necessary estimates. When the two index parameters, αs, can be held to be identical, but may be still unknown, we propose a quantile-based classification rule by exploiting a convolution type property and some results on the tail behavior of stable distributions. Fisher type rules are described for the general case. Details of the computer programs with necessary source codes are provided to enable the user to implement our rules for real-life data sets. An example based on a real-life data set provided to us by Rachev is also given.

Keywords—α-stable distribution, Apparent error rate, Classification rule, Quantile-based discrimination, Tail probability.

1. INTRODUCTION AND SUMMARY

We consider the problem of classification of an observation from one of two strictly α-stable populations ($\alpha \neq 1$) with unknown parameters, based on training samples, one each from the two populations. The case of equal αs is first considered in Section 2. A simple and elegant quantile-based rule is constructed by exploiting a well-known property of the difference of two stable random variables and also by using a closed form expression for the tail probability of a stable distribution. Such rules are often needed in practice, e.g., to identify, based on a new observation, whether the financial market for a commodity has entered a "bullish" or a "bearish" characteristic, as has been classified, and thus, the corresponding parameters also determined, by

The authors are extremely grateful to Prof. S. T. Rachev for his continuous encouragement to work on this problem, for his valuable constructive criticisms, for the program on estimation of the parameters of a stable distribution, and finally for the real-life data set used in this paper. The first author would like to thank University of California-Santa Barbara for his Visiting Professorship there, which led to this work. The second author is indebted to Council of Scientific and Industrial Research, Government of India, for offering him a Senior Research Fellowship and to Indian Statistical Institute, where most of this work was done as a part of his Ph.D. dissertation.

0895-7177/01/$ - see front matter Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII: S0895-7177(01)00117-0

the training samples from such populations provided earlier. A detailed analysis, including error rates, of an illustrative example based on a real-life data set provided to us by Rachev is given in Section 3. In Section 4, we describe Fisher type rules for the general situation. Section 5 briefly describes the computational support called for. Details of the computer programs with necessary source codes are provided in the Appendix to facilitate the implementation of our proposed rules in practice. Further details and the smaller subroutines not included here, if needed, may be available to the interested readers from the authors.

We believe that this work is the very first one on statistical classification for stable distributions and will pave the road for future research in this area by exposing interesting problems.

We require the following adaptations from results in [1].

THEOREM 1.1. *For an α-stable ($\alpha \neq 1$) random variable Y with admissible parameter quadruple $(\alpha, \beta, \gamma, \lambda)$, there exists a unique representation such that*

$$Y(\alpha, \beta, \gamma, \lambda) \stackrel{d}{=} \lambda^{1/\alpha} Y(\alpha, \beta, 0, 1) + \lambda\gamma. \tag{1.1}$$

THEOREM 1.2. *Let $Y(\alpha, \beta_1, \gamma_1, \lambda_1)$ and $Y(\alpha, \beta_2, \gamma_2, \lambda_2)$ be two independent α-stable random variables. Then $Y(\alpha, \beta_1, \gamma_1, \lambda_1) - Y(\alpha, \beta_2, \gamma_2, \lambda_2)$ is distributed as $Y(\alpha, \beta, \gamma, \lambda)$ where $\lambda = \lambda_1 + \lambda_2$, $\lambda\beta = \lambda_1\beta_1 - \lambda_2\beta_2$, and $\lambda\gamma = \lambda_1\gamma_1 - \lambda_2\gamma_2$.*

2. CONSTRUCTION OF THE CLASSIFICATION RULE

Let

$$d_{ij} = Y(\alpha, \beta_i, \gamma_i, \lambda_i) - Y(\alpha, \beta_j, \gamma_j, \lambda_j). \tag{2.1}$$

Therefore,

$$d_{ii} = Y(\alpha, 0, 0, 2\lambda_i), \tag{2.2}$$

$$d_{ij} = Y\left(\alpha, \frac{\lambda_i\beta_i - \lambda_j\beta_j}{\lambda_1 + \lambda_2}, \frac{\lambda_i\gamma_i - \lambda_j\gamma_j}{\lambda_1 + \lambda_2}, \lambda_1 + \lambda_2\right), \qquad i \neq j. \tag{2.3}$$

For $\alpha \neq 1$, the tail probability of a standard stable distribution at 0, $F(0)$ has a closed form expression; i.e.,

$$F(0) = 1 - \frac{1}{2}\left(1 - \beta\frac{K(\alpha)}{\alpha}\right). \tag{2.4}$$

Let Y_is be strictly stable. Let $\beta = (\lambda_1\beta_1 - \lambda_2\beta_2)/(\lambda_1 + \lambda_2)$. Then by (2.2),

$$d_{ii} = (2\lambda_i)^{1/\alpha} Y(\alpha, 0, 0, 1), \qquad d_{ij} = (\lambda_1 + \lambda_2)^{1/\alpha} Y(\alpha, (j-i)\beta, 0, 1).$$

Therefore,

$$P(d_{ii} > 0) = P(Y(\alpha, 0, 0, 1) > 0) = \frac{1}{2}.$$

Let $p_{ij} = P(d_{ij} > 0)$. Then

$$p_{22} = \frac{1}{2} = p_{11}, \tag{2.5}$$

$$P(d_{ij} > 0) = \frac{1}{2}\left(1 + (j-i)\beta\frac{K(\alpha)}{\alpha}\right) = p_{ij}. \tag{2.6}$$

Note that $p_{12} + p_{21} = 1$.

2.1. The Statistic $G_{n,i}$

Let, for a given real r.v. Z, $1_{[Z]}$ be the indicator function of the event $\{Z > 0\}$. Let $\{X_{ik}, i = 1,2;\ k = 1,\ldots,n_i\}$ be two samples from strictly α-stable distributions—with admissible parameter quadruples $(\alpha, \beta_i, 0, \lambda_i)$, $i = 1,2$. Let Y be a new observation (unclassified) from one of the two given populations. Define

$$G_{n,i} = \frac{1}{n_i} \sum_{k=1}^{n_i} 1_{[X_{ik} - Y_{n+1}]}. \tag{2.7}$$

The classification rule can now be formulated as follows:

$$\text{if } \left|\frac{1}{2} - G_{n,1}\right| < \left|\frac{1}{2} - G_{n,2}\right| \text{ assign to Population 1, o.w. assign to Population 2.}$$

A corresponding modification to account for inclusion probabilities is:

$$\text{if } \pi_1 \left|\frac{1}{2} - G_{n,1}\right| < \pi_2 \left|\frac{1}{2} - G_{n,2}\right| \text{ assign to Population 1, o.w. assign to Population 2.}$$

REMARK 2.1. At this point, although nothing can be said about nonstrictly stable distributions in general, a few simple manipulations, with the integral representations of the distribution functions of stable distributions as given by Zolotarev, show that the above procedure can be applied to the nonstrictly stable distributions given by $\alpha < 1$, $\beta < 1$, and $\alpha > 1$, i.e., the classes for which $p_{12} \neq 0.5$.

2.2. The Distribution of the $G_{n,i}$

From (2.7), we have that each indicator in the defining equation follows a binomial distribution, taking the value 1 with probability p, where

$$p = E\left(1_{[X_{i1} - Y]}\right), \qquad i = 1,2. \tag{2.8}$$

Note that p takes one of the values p_{ij}s.

The exact distribution is found out by considering the distribution of the number of X_{ik}s that are greater than Y.

$$P(n_i G_{n,i} = k) = \binom{n_i}{k} p^k (1-p)^{n_i - k}, \tag{2.9}$$

$$P\left(\left|G_{n,i} - \frac{1}{2}\right| = h\right) = \binom{n_i}{j} \left(p^k (1-p)^{n_i - k} + (1-p)^k p^{n_i - k}\right), \qquad k = \left[\frac{n_i}{2}\right] + j. \tag{2.10}$$

3. A REAL LIFE EXAMPLE

3.1. Description of the Data Set

The original data set provided to us by Rachev (who should be contacted for any further details of this data set) consisted of share prices of a certain category, over a period of time, which are transformed by taking sequential differences of natural logarithms of each share price as is usually done in econometric studies of price phenomena (see [2]). The justification for dividing the set into two samples is on the basis of a "changepoint" from "bullish" to "bearish" tendencies.

3.2. Construction of the Rule

A brief description of the SPLUS functions and program parts used is given below. Detailed listings of these programs are available to the interested readers from the authors.

Total samples size is 614. `nsamp1` gives the breakpoint at which the sample is partitioned. This is set to be 320. `prlog` is the transformed data set which is created by taking the successive differences of the log of the input vector of raw data.

```
f <- function(u,sam,nsam)
{q <- length(sam[sam > u])/nsam
q}
```

This actually gives the empirical tail probability at `u` based on sample `sam`. `p1`, `p2` are estimates of population inclusion probabilities.

```
fq <- function(x,d1,d2)
{q1 <- f(x,samp1,nsamp1)
q2 <- f(x,samp2,nsamp2)
val <- (d1*abs(q1-0.5)) - (d2*abs(q2-0.5))
val}
```

The function `fq` actually constructs the rule which is of the form

$$r(x) = d1 \cdot \left| q_1(x) - \frac{1}{2} \right| - d2. \left| q_2(x) - \frac{1}{2} \right|. \tag{3.1}$$

Figure 1 shows a comparison of the constructed rule with and without considering inclusion probabilities. The lighter line shows without and the darker one shows the rule with inclusion probabilities.

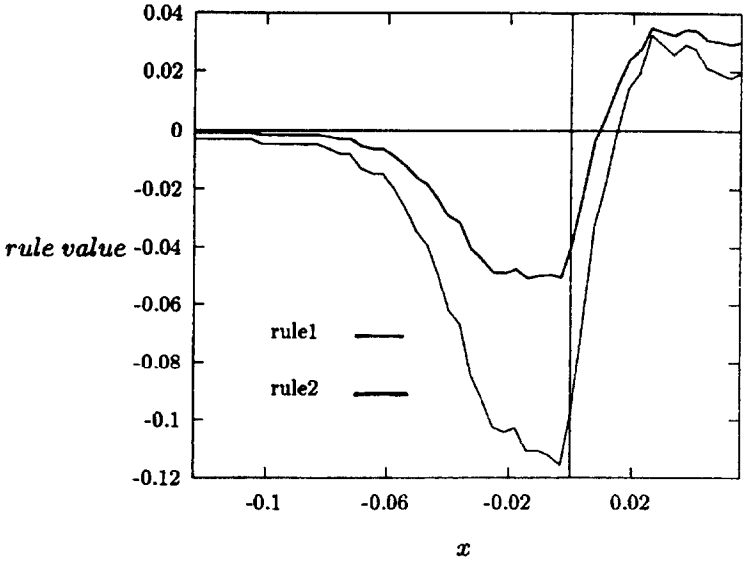

Figure 1. Plot of two rules.

Rough estimates were obtained by considering the upper extreme $1/10^{\text{th}}$ of each sample. Apparent error rates (AERs) were obtained using `esterr` function. This function first computes samplewise error estimates `prop1`, `prop2` and then combines them to obtain the combined sample AER (`prop3`).

3.3. Further Comparison Against Sample Homogeneity

Incorporating a C-routine supplied by Rachev [3] to estimate the parameter quadruples of a stable distribution, a C-routine developed by the authors calculates the AER obtained from the

rule constructed, and the corresponding parameter estimates for the assumed underlying stable populations, given the choice of subsamples (input by the user).

Rachev's estimator fails for certain ranges, for which however, the rule can still be constructed but no comparisons can be made as to sample homogeneity. Incidentally, some of these regions appear to give the lowest AERs.

Varying these subsample choices, we report the following: c is the transformed "scale" factor ($\lambda^{1/\alpha}$ of Zolotarev [1]) and δ is the transformed "shift", as in [3].

First Pair of Subsamples.

subsample-1 lower limit = 288 (i.e., starting from data no. 288 of original set), upper limit = 429

subsample-2 lower limit = 430, upper limit = 500

error estimate for 1st sample = 0.414286, for 2nd sample = 0.528571, for combined sample = 0.312796.

parameter estimates for subsample-1

$\alpha = 1.694558$, $\beta = 0.227588$, $c = 0.021124$, $\delta = -0.006073$.

parameter estimates for subsample-2

$\alpha = 1.641819$, $\beta = -0.094336$, $c = 0.017767$, $\delta = -0.002327$.

Second Pair of Subsamples.

subsample-1 lower limit = 250, upper limit = 450

subsample-2 lower limit = 451, upper limit = 600

error estimate for 1st sample = 0.483221, for 2nd sample = 0.469799, for combined sample = 0.406877.

parameter estimates for subsample-1

$\alpha = 1.585061$, $\beta = -0.191576$, $c = 0.017243$, $\delta = -0.003982$.

parameter estimates for subsample-2

$\alpha = 1.637159$, $\beta = -0.270756$, $c = 0.016563$, $\delta = 0.001094$.

3.4. Conclusions

Shifting the breakpoint back from 320 to 200 reduces the AER to about 0.33, which combined with the corresponding graphical comparison of the indices of the two subsamples indicates that retrogression of the breakpoint reduces the index heterogeneity between the two groups, and hence, the rule performs better.

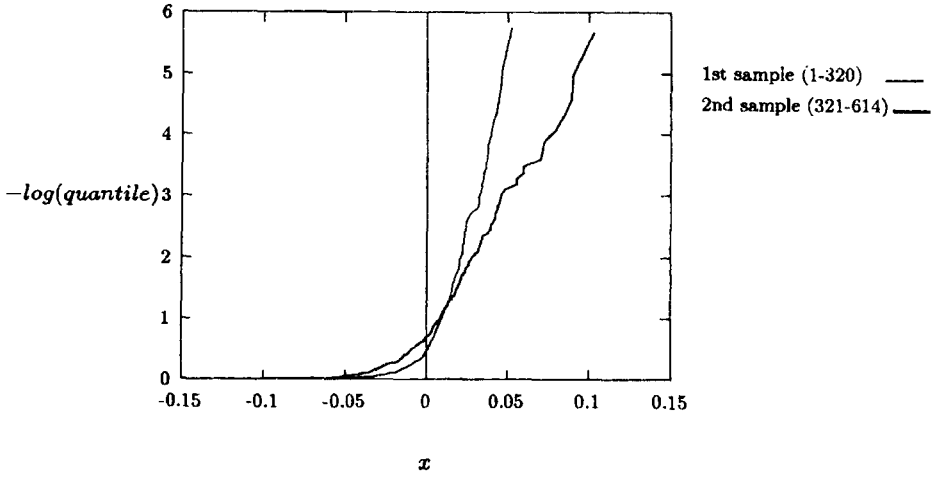

Figure 2. Plot of $-\log$(quantile) for two samples.

REMARKS 3.1. The above analysis inherently assumes the homogeneity of the index parameters of the two stable distributions. From a well-known result (see [4]), the negative of the log-tail asymptotically goes to the index α of the underlying α-stable distribution. We exploit this result to produce Figure 2, which shows a comparison of the log-tails of the two empirical distribution functions. The lighter line shows the first sample, and the darker one shows the second sample. The plot suggests the possibility of a difference in the indices of the two distributions, indicating thereby that the initial breakpoint to separate the two samples from the original dataset is not quite justified in taking the two samples to be homogeneous with respect to the index α. For our example above, we have avoided this problem by judiciously selecting the breakpoint. However, in other real-life problems, one may naturally encounter two differing index parameters. A generalization (nontrivial) then of our above approach to cover such situations is, thus, of practical necessity and importance—a problem we intend to take up for future research. An alternative, however computationally involved, approach is given below.

4. FISHER TYPE DISCRIMINATION RULES

Since, in general, closed form expressions for the stable densities in terms of elementary functions are not available, it is difficult to visualize density ratio type discriminant functions. However, Zolotarev [1] gives an analytical form containing definite integrals, which can be used as a basis for constructing approximations to the actual density ratios. This method, though computation intensive, suffers from theoretical difficulties—notably, absence of conclusive results about the existence of expectations of derivatives of log-densities of stable distributions.

From the computational viewpoint, using the generalized pseudo-maximum likelihood (GPML) framework as developed by us (SenGupta and Roy [5]), one can also incorporate this density-ratio in a logistic discrimination-type (SenGupta and Roy [6]) GPML framework.

4.1. Representation of Stable Densities

The following representations are from Zolotarev [1].

THEOREM 4.1. *Let*

$$\epsilon(\alpha) = \operatorname{sgn}(1-\alpha), \qquad \theta = \beta\frac{K(\alpha)}{\alpha}, \qquad \text{and} \qquad \theta^\star = \theta \operatorname{sgn} x.$$

Let

$$U_\alpha(\phi,\theta) = \left(\frac{\sin(\pi\alpha/2)(\phi+\theta)}{\cos(\pi\phi/2)}\right)^{\alpha/(1-\alpha)} \cdot \frac{\cos(\pi/2)\left((\alpha-1)\phi+\alpha\theta\right)}{\cos(\pi\phi/2)},$$

$$U_1(\phi,\beta) = \frac{\pi(1+\beta\phi)}{2\cos(\pi\phi/2)} \cdot \exp\left(\frac{\pi}{2}\left(\phi+\frac{1}{\beta}\right)\tan\frac{\pi\phi}{2}\right).$$

Then the densities of standard stable distributions can be written as follows.

(1) *If* $\alpha \neq 1$, $|\beta| \leq 1$, $x \neq 0$, *then*

$$g(x,\alpha,\beta) = \frac{\alpha|x|^{1/(\alpha-1)}}{2|1-\alpha|} \cdot \int_{-\theta^\star}^{1} U_\alpha(\phi,\theta^\star) \exp\left(-|x|^{\alpha/(\alpha-1)} U_\alpha(\phi,\theta^\star)\right) d\phi. \tag{4.1}$$

(2) *If* $\alpha = 1$, $|\beta| \neq 0$, $\forall\, x$,

$$g(x,1,\beta) = \frac{e^{-x/\beta}}{2|\beta|} \cdot \int_{-1}^{1} U_1(\phi,\beta) \exp\left(-e^{-x/\beta} U_1(\phi,\beta)\right) d\phi. \tag{4.2}$$

(3) *If* $\alpha \neq 1$, $x = 0$, *then*

$$g(0,\alpha,\beta) = \frac{1}{\pi}\Gamma\left(1+\frac{1}{\alpha}\cos\left(\frac{\pi\beta K(\alpha)}{\alpha}\right)\right). \tag{4.3}$$

(4) $\alpha = 1$, $\beta = 0$ *corresponds to the Cauchy distribution.*

4.2. Construction of the Discrimination Rules

Let $h(x;\alpha_1,\beta_1,\alpha_2,\beta_2) \equiv h(x;\underline{\alpha},\underline{\beta}) \equiv h(x)$,

$$h(x) = \frac{g(x,\alpha_1,\beta_1)}{g(x,\alpha_2,\beta_2)}$$

$$\underline{\alpha} = \begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} \quad \underline{\beta} = \begin{pmatrix}\beta_1\\ \beta_2\end{pmatrix} \quad \underline{b} = \begin{pmatrix}b\\ b\end{pmatrix}, \qquad b \in \mathbb{R}.$$

We first consider standard stable distributions and classify them into three major groups by the form of their density representations under $h(x)$ within these groups. Next, we replace the integrals in the numerator and denominator by composite quadrature formulae, i.e., divide the range of integration into several subintervals and apply a simple quadrature formula to each of them.

If the underlying distributional parameters are already known, we simply substitute these values in the computed ratios, or use estimated values of the parameters from the sample derived from either empirical c.f.s or other methods.

Since the quadrature formulae occurring in the numerator and denominator can be computed with arbitrary accuracy, we have a simple though computationally intensive method of discrimination in stable distributions. Thus, it only remains to present $h(x)$ explicitly.

Denote

(1) for $\alpha \neq 1$, $x \neq 0$, $|\beta| \leq 1$,

$$V_\alpha(\phi) = U_\alpha(\phi,\theta^\star)\exp\left(-|x|^{\alpha/(\alpha-1)}U_\alpha(\phi,\theta^\star)\right), \tag{4.4}$$

where $-\theta^\star \leq \phi \leq 1$;

(2) for $\alpha = 1$, $\beta \neq 0$, $\forall x$,

$$V_1(\phi) = U_1(\phi,\beta)\exp\left(-e^{-x/\beta}U_1(\phi,\beta)\right). \tag{4.5}$$

We sometimes use the notation V_α or V_1 for short, equivalently.

Denote

$$I_{\alpha,\theta^\star}(x) \equiv \int_{-\theta^\star}^{1} V_\alpha(\phi)\,d\phi, \qquad I_{1,\beta}(x) \equiv \int_{-1}^{1} V_1(\phi)\,d\phi.$$

We now present $h(x)$ explicitly. Note that several cases are possible.

(1) $\underline{\alpha} \neq \underline{1}$, $x \neq 0$, $|\beta_1| \leq 1$, $|\beta_2| \leq 1$,

$$h_1(x) = \begin{cases} \dfrac{\alpha_1|1-\alpha_2|}{\alpha_2|1-\alpha_1|}\,|x|^{\alpha_1/(\alpha_1-1)-\alpha_2/(\alpha_2-1)} \cdot \dfrac{I_{\alpha_1,\theta_1^\star}(x)}{I_{\alpha_2,\theta_2^\star}(x)}, & \text{if } x \neq 0,\\ \dfrac{\Gamma(1+1/\alpha_1)\cos((\pi/2)\,\theta_1)}{\Gamma(1+1/\alpha_2)\cos((\pi/2)\,\theta_2)}, & \text{if } x = 0. \end{cases} \tag{4.6}$$

(2) $\underline{\alpha} = \underline{1}$, $\underline{\beta} \neq \underline{0}$, $\forall x$,

$$h_2(x) = \frac{|\beta_2|}{|\beta_1|}\,e^{-x(1/\beta_1-1/\beta_2)} \cdot \frac{I_{1,\beta_1}(x)}{I_{1,\beta_2}(x)}. \tag{4.7}$$

(3) $\alpha_1 \neq 1$, $\alpha_2 = 1$, $|\beta_1| \leq 1$, $\beta_2 \neq 0$,

$$h_3(x) = \frac{\alpha_1|\beta_2|}{|1-\alpha_1|} \cdot |x|^{1/(\alpha_1-1)} \cdot e^{x/\beta_2}\,\frac{I_{\alpha_1,\theta_1^\star}(x)}{I_{1,\beta_2}(x)}, \qquad \text{if } x \neq 0, \tag{4.8}$$

$$= \frac{1}{2\pi|\beta_2|}\,\Gamma\left(1+\frac{1}{\alpha_1}\right)\cos\left(\frac{\pi}{2}\,\theta_1\right)\frac{1}{I_{1,\beta_2}(0)}, \qquad \text{if } x = 0. \tag{4.9}$$

REMARKS 4.1. We observe that by using the form of the ratio of the densities, and estimating in the usual GPML iterative schemes, we can arrive at estimators of stable parameter quadruples as well as the rule simultaneously.

5. PROGRAMS AND COMPUTATIONAL SUPPORT

Almost all the computer programs which have been developed for the specific purpose of computational work required in this work has been written in C and compiled in TURBO C. Apart from programs written in Turbo-C, the statistical package SPLUS (Windows Version 3.3) has been used. Graphs have been drawn using GNUPLOT 2.2. The GNUPLOT programs here have been designed to output LaTeX output which need to be included in a LaTeX document. Commenting out the set terminal and set output options in the programs will generate screen outputs.

A contents-list of the major programs used in this paper which may be used for implementation of our rules in classification of other qualifying real-life data sets is given below. The programs have been detailed with the appropriate source codes in the Appendix. Listing of supporting smaller files called by these programs are available to the interested readers from the authors.

1. **combin1.c** combin1.c is a C-routine that incorporates the program to estimate stable parameter quadruples as also the corresponding discrimination rule based on quantiles.
2. **stabfig.c** The C-program that generates tables for the GNUPLOT programs "stab1.inp" and "stab2.inp" for stable discrimination.
3. **stab1.inp** GNUPLOT program to plot the two rules, i.e., with and without considering inclusion probabilities.
4. **stab2.inp** GNUPLOT program to plot the estimated log-quantiles for the two subsamples.

6. FIGURES

The figures shown here were all drawn using GNUPLOT programs which can be used to produce LaTeX files. The first plot compares the quantile-based rules with and without considering inclusion probabilities. The second plot shows that the initial breakpoint to separate the two samples from the original dataset is not quite justified in taking the two samples to be homogeneous with respect to the index or α.

APPENDIX

combin1.c

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<time.h>

main(argc,argv)
int argc;
char *argv[];
{ FILE *datfile;
 double *stk, *stock,*samp[2],*rnstab,q[2];
double *param,*prop,*est_par,ind,x,val,*test_smp[2];
float temp;
int i,j,k,count,cnt,lim,range;
          int *limits[2],length;
          char *data;
void estimate(int smp_no, double *stk,int range,
double *est_par);

void stabgen(double *rnstab,double *par,int length);

void error_calc(double *samp[2],
  double *test[2],int *limits[2],
```

```
int *limits1[2], double *prop, double *param);

void err_print(int argc,char *argv[],double *prop,
int *limits[2]);

void est_print(int argc,char *argv[],
  double *est_par,int j);

double absol(double x, double y);
double square(double x, double y);

    data = argv[1];
    if (data == NULL)
    {
       printf("combin [datafile] [outfile]\n");
    exit(0);
       }
     datfile = fopen(data,"r");
    if (datfile == NULL)
    {
    printf("No such file exists. Try again.\n");
    exit(0);
       }
    count=0;
    while( fscanf(datfile,"%f",&temp) != EOF)
    {count++;}
    rewind(datfile);
    stock = (double *)malloc(count*sizeof(double));
    for(i=0;i<count;i++)
    { fscanf(datfile,"%f",&temp);
     *(stock+i) = (double) temp;
      }

 printf("\nTotal no. of data points in file = %d\n",count);
for(i=0;i<2;i++){
limits[i] = (int *)malloc(3*sizeof(int));
}
*limits[0]=*limits[1]=1;
*(limits[0]+1)=*(limits[1]+1)=count;
if(argc > 2)
{
printf("\nType 1-st sample starting lower limit\n");
scanf("%d",&k);
*limits[0]=k;
printf("\nType 1-st sample end, upper limit\n");
scanf("%d",&k);
*(limits[0]+1)=k;
printf("\nType 2-nd sample starting lower limit\n");
scanf("%d",&k);
*limits[1]=k;
printf("\nType 2-nd sample end, upper limit\n");
scanf("%d",&k);
*(limits[1]+1)=k;
};

for(j=0;j<2;j++){
*(limits[j]+2)=*(limits[j]+1)-*limits[j];
samp[j] = (double *) malloc(*(limits[j]+2)*sizeof(double));
```

```
}
for(j=0;j<2;j++)
{
for(i=0;i< *(limits[j]+2);i++)
{ *(samp[j]+i)=log(*(stock+i+1+ *limits[j]))
  -log(*(stock+i+ *limits[j]));}
}
fclose(datfile);

param = (double *)malloc(3*sizeof(double));
*param=0.5;
*(param+1)=1.0;
*(param+2)=1.0;

/* param pints to (0) - pivot value (default 0.5)
                  (1) - scale factor of 1-st component
                  (2) - scale factor of 2-nd component*/
prop= (double *)malloc(3*sizeof(double));
*(prop+2)=0.0;
error_calc(samp,samp,limits,limits,prop,param);
err_print(argc,argv,prop,limits);
est_par = (double *)malloc(4*sizeof(double));

for(j=0;j<2;j++)
{
range=*(limits[j]+2);
stk=(double *)malloc(range*sizeof(double));

for(i=0;i<range;i++){*(stk+i)=*(samp[j]+i);}
estimate(j,stk,range,est_par);
est_print(argc,argv,est_par,j);

free(stk);
length=range;
rnstab = (double *)malloc(length*sizeof(double));
stabgen(rnstab,est_par,length);
estimate(j+2,rnstab,length,est_par);

test_smp[j] = (double *)malloc(length*sizeof(double));
for(i=0;i<length;i++){*(test_smp[j]+i)=*(rnstab+i);}
free(rnstab);
est_print(argc,argv,est_par,j+2);
}

error_calc(samp,test_smp,limits,limits,prop,param);
err_print(argc,argv,prop,limits);
}

void err_print(int argc,char *argv[],double *prop,
int *limits[2])
{ FILE *outfile;
printf("\n");
     printf("error estimate for
 1-st sample = %lf\n",*prop);
     printf("error estimate for
2-nd sample = %lf\n",*(prop+1));
         printf("error estimate for
combined sample = %lf\n",*(prop+2));
```

```
     printf("sample_1 lower limit = %d\n",*limits[0]);
 printf("sample_1 upper limit = %d\n",*(limits[0]+1));
 printf("sample_2 lower limit = %d\n",*limits[1]);
 printf("sample_2 upper limit = %d\n",*(limits[1]+1));

 if(argc == 3)
 { outfile = fopen(argv[2],"a");
     fprintf(outfile,"\n");
     fprintf(outfile,"error estimate for
 1-st sample = %lf\n",*prop);
     fprintf(outfile,"error estimate for
 2-nd sample = %lf\n",*(prop+1));
     fprintf(outfile,"error estimate for
 combined sample = %lf\n",*(prop+2));
     fprintf(outfile,"sample_1 lower limit
 = %d\n",*limits[0]);
 fprintf(outfile,"sample_1 upper limit
 = %d\n",*(limits[0]+1));
 fprintf(outfile,"sample_2 lower limit
 = %d\n",*limits[1]);
 fprintf(outfile,"sample_2 upper limit
 = %d\n",*(limits[1]+1));
 fclose(outfile);
 };
 }

 void est_print(int argc,char *argv[],
 double *est_par,int j)
 {FILE *outfile;
  printf("\n parameter estimates for sample_%d\n",j);
     printf("alpha = %lf\n",*est_par);
     printf("beta  = %lf\n",*(est_par+1));
   printf("c     = %lf\n",*(est_par+2));
     printf("delta = %lf\n",*(est_par+3));

 if(argc == 3)
   { outfile = fopen(argv[2],"a");
     fprintf(outfile,"\n parameter estimates
 for sample_%d\n",j);
     fprintf(outfile,"alpha = %lf\n",*est_par);
     fprintf(outfile,"beta  = %lf\n",*(est_par+1));
   fprintf(outfile,"c     = %lf\n",*(est_par+2));
     fprintf(outfile,"delta = %lf\n",*(est_par+3));
    fclose(outfile);
   };
 }

 void error_calc(double *samp[2],
  double *test[2],int *limits[2],
 int *limits1[2],double *prop, double *param)
 {int j,i,jj,ii,cnt,lim,lim1;

 double q[2],x,ind,level=0.0,val;
 double absol(double x, double y);

 for(j=0;j<2;j++)
 {
 ind = (double) (1-(2*j));
```

```
cnt=0;lim=*(limits1[j]+2);
for(i=0;i<lim;i++)
  { x=*(test[j]+i);

for(jj=0;jj<2;jj++){
q[jj]=0; lim1=*(limits[jj]+2);
 for(ii=0;ii<lim1;ii++)
 {if(x > *(samp[jj]+ii)){q[jj] +=1.0;};
     }
  q[jj] /= (double) lim1;
  }
 val = absol(q[0],*param)-absol(q[1],*param);
if(ind*(val-level) > 0.0)
{cnt++;};}
*(prop+j)= ((double) cnt)/((double) lim);
*(prop+2) += (double) cnt;
}
*(prop+2) /= (double) (*(limits[0]+2)+ *(limits[1]+2));
}

double absol(double x, double y)
{return(fabs(x-y));}

double square(double x, double y)
{return( (x-y)*(x-y));}

void stabgen(double *rnstab,double *par,int length)
{FILE *rstab;
int i;
 double u1,u2,v1,v2,v3,w,phi,phi0;
 double kalpha,val,pi_by2,pi=3.14159;
  pi_by2=pi/2.0;
   randomize();
kalpha=*par;
     if(kalpha > 1.0){kalpha -= 2.0;};
     phi0=(-1.0)*pi_by2* *(par+1) *kalpha/(*par);
rstab= fopen("stabvar","a");
if(*par != 1.0)
{for(i=0;i<length;i++)
   {
u1= rand();
u1 /= (double) 32767;
u2= rand();
u2 /= (double) 32767;
   w=(-1.0)*log(u1);
    phi=pi*(u2-0.5);
       val=*par *(phi-phi0);
  v1 = cos(phi-val)/w;
   v2 = sin(val);
    v3 = cos(phi);
   val = exp( (log(v1*v3))/(*par));
     val /= v1; val *=v2;
   val *= *(par+2);
 val += *(par+3)/(*(par+2));
     *(rnstab+i)=val;
       fprintf(rstab,"\n *rnstab = %lf\n",*(rnstab+i));
     }
```

```
 };
if(*par == 1.0)
{
for(i=0;i<length;i++)
   {
   u1= rand();
   u1 /= (double) 32767;
   u2= rand();
   u2/= (double) 32767;
    w=(-1.0)*log(u1);
     phi=pi*(u2-0.5);
      v1=2.0* *(par+1)/pi;
       v2= 1.0+(v1*phi);
  *(rnstab+i)=(v2*tan(phi))-(v1*log(w*cos(phi)/v2));
    }
};
}

void estimate(int smp_no, double *stk,
 int range,double *est_par)
{     double x_95,x_75,x_50,x_25,x_05;
      int i_95,i_75,i_50,i_25,i_05;
   double na,nb,nc,nd,x;
   float temp;
   int i,j,k,count;
   double a,b,c,d,*aa;
int mycompare(double *x, double *y);
double interpolate(double, double,
  double,double,double);

void table_1(double x, double y, double *aa);
double table_2(double, double);
double table_3(double, double);

aa = (double *) malloc(2*sizeof(double));
      count=range;
     qsort(stk,count,sizeof(double),mycompare);
    x = (double) (2.0*count*.05+1.0)*.5;
     i_05 = (int) x;
   x_05 = interpolate((double)i_05,*(stk+i_05-1),
        (double)i_05+1,*(stk+i_05),x);
          x = (double) (2.0*count*.25+1.0)*.5;
      i_25 = (int) x;
x_25 = interpolate((double)i_25,*(stk+i_25-1),
(double)i_25+1,*(stk+i_25),x);
x = (double) (2.0*count*.5+1.0)*.5;
     i_50 = (int) x;
x_50 = interpolate((double)i_50,*(stk+i_50-1),
 (double)i_50+1,*(stk+i_50),x);
     x = (double) (2.0*count*.75+1.0)*.5;
           i_75 = (int)x;
x_75 = interpolate((double)i_75,*(stk+i_75-1),
(double)i_75+1,*(stk+i_75),x);
   x  = (double) (2.0*count*.95+1.0)*.5;
      i_95 = (int) x;
      x_95 = interpolate((double)i_95,*(stk+i_95-1),
```

```
       (double)i_95+1,*(stk+i_95),x);
 na = (x_95 - x_05)/(x_75 - x_25);
      nb = (x_95 + x_05 - 2.0*x_50)/(x_95 - x_05);
      table_1(na,nb,aa);
      a=*aa; b=*(aa+1);
      nc = table_2(a,b);
     c = (x_75 - x_25)/nc;
      nd = table_3(a,b);
      d = x_50 + c*nd - b*c*tan(3.14159265*a/2.0);
  printf("\n parameter estimates for sample_%d\n",smp_no);
     printf("alpha = %lf\n",a);
     printf("beta  = %lf\n",b);
  printf("c     = %lf\n",c);
     printf("delta = %lf\n",d);
*est_par=a;
*(est_par+1)=b;
*(est_par+2)=c;
*(est_par+3)=d;
}

int mycompare (double *x,double *y)
{
     if ( (*x - *y) > 0 ) {return(1);}
   else {if ((*x -*y) <0) return (-1);};
   return(0);
     }
double interpolate(x1,y1,x2,y2,x)
double x1,y1,x2,y2,x;
{     double tmp;
      tmp = y1 + (x- x1)*(y2 - y1)/(x2 - x1);
    return(tmp);}

void table_1( double x, double y, double *aa)
{
       int i,j,pt_x=14,pt_y=6;
       double a_1,a_2,b_1,b_2,a,b;

double data_1[] = {2.0    ,2.0    ,2.0     ,2.0    ,
2.0    ,2.0   ,2.0    ,
1.916  ,1.924  ,1.924   ,1.924  ,1.924  ,1.924 ,1.924  ,
1.808  ,1.813  ,1.829   ,1.829  ,1.829  ,1.829 ,1.829  ,
1.729  ,1.730  ,1.737   ,1.745  ,1.745  ,1.745 ,1.745  ,
1.664  ,1.663  ,1.663   ,1.668  ,1.676  ,1.676 ,1.676  ,
1.563  ,1.560  ,1.553   ,1.548  ,1.547  ,1.547 ,1.547  ,
1.484  ,1.480  ,1.471   ,1.460  ,1.448  ,1.438 ,1.438  ,
1.391  ,1.386  ,1.378   ,1.364  ,1.337  ,1.318 ,1.318  ,
1.279  ,1.273  ,1.266   ,1.250  ,1.210  ,1.184 ,1.150  ,
1.128  ,1.121  ,1.114   ,1.101  ,1.067  ,1.027 ,0.973  ,
1.029  ,1.021  ,1.014   ,1.004  ,0.974  ,0.935 , .874  ,
.896   , .892   , .887   , .883  , .855  , .823 , .769  ,
.818   , .812   , .806   , .801  , .780  , .756 , .691  ,
.698   , .695   , .692   , .689  , .676  , .656 , .595  ,
.593   , .590   , .588   , .586  , .579  , .563 , .513  };

double data_2[] = {0.0,2.160,1.0,1.0,1.0,1.0,1.0     ,
     0.0,1.592,3.39,1.0,1.0,1.0,1.0     ,
 0.0     , .759  ,1.8     ,1.0    ,1.0    ,1.0   ,1.0     ,
```

```
 0.0    , .482  ,1.048   ,1.694  ,1.0    ,1.0   ,1.0    ,
 0.0    , .360  , .76    ,1.232  ,2.229  ,1.0   ,1.0    ,
 0.0    , .253  , .518   , .823  ,1.575  ,1.0   ,1.0    ,
 0.0    , .203  , .41    , .632  ,1.244  ,1.906 ,1.0    ,
 0.0    , .165  , .332   , .499  , .943  ,1.56  ,1.0    ,
 0.0    , .136  , .271   , .404  , .689  ,1.23  ,2.195  ,
 0.0    , .109  , .216   , .323  , .539  , .827 ,1.917  ,
 0.0    , .096  , .19    , .284  , .472  , .693 ,1.759  ,
 0.0    , .082  , .163   , .243  , .412  , .601 ,1.596  ,
 0.0    , .074  , .147   , .22   , .377  , .546 ,1.482  ,
 0.0    , .064  , .128   , .191  , .33   , .478 ,1.362  ,
 0.0    , .056  , .112   , .167  , .285  , .428 ,1.274  };
double nu_x[] = {2.439,2.5,2.6,2.7,2.8,3.0,3.2,
   3.5,4.0,5.0,6.0,8.0,10.0,15.0,25.0};
double nu_y[] = {0.0,0.1,0.2,0.3,0.5,0.7,1.0};

       if (x < 2.439)
       {  a = 2.0;    b = y; return;}
       else if ( fabs(y) > 1.0)
       {
    b = (y/fabs(y));
        i=0;
    while(i < 15)
        {
  if (x < nu_x[i])
  { pt_x = i-1;
     break;
     }
 i++;       }
  a = interpolate(nu_x[pt_x],data_1[7*pt_x+6],
       nu_x[pt_x +1],
    data_1[7*pt_x+13],x);
    return;
 }
      else
 {  i=0;
     while (i<15)
     {
         if (x < nu_x[i])
         {
    pt_x = i-1;
    break;
        }
        i++;
     }
     j=0;
     while (j< 7)
     {
        if (fabs(y) < nu_y[j])
        {   pt_y = j-1;
     break;
        }
        j++;
     }
a_1=interpolate(nu_x[pt_x],data_1[7*pt_x+pt_y],
```

```
  nu_x[pt_x+1],data_1[7*pt_x+pt_y+7],x);
a_2=interpolate(nu_x[pt_x],data_1[7*pt_x+pt_y+1],
  nu_x[pt_x+1],data_1[7*pt_x+pt_y+8],x);
a=interpolate(nu_y[pt_y],a_1,nu_y[pt_y+1],a_2,fabs(y));
b_1=interpolate(nu_x[pt_x],data_2[7*pt_x+pt_y],
  nu_x[pt_x+1],data_2[7*pt_x+pt_y+7],x);
  b_2=interpolate(nu_x[pt_x],data_2[7*pt_x+pt_y+1],
  nu_x[pt_x+1],data_2[7*pt_x+pt_y+8],x);
b=interpolate(nu_y[pt_y],b_1,nu_y[pt_y+1],b_2,fabs(y));
        if (b > 1.0) b =1.0;
        b = (y/fabs(y))* b;
        *aa=a; *(aa+1)=b;
        }}
double table_2(x,y)
double x,y;
{
      int i,j,pt_x=15,pt_y=4;
      double nc1,nc2,nc3;
double data[] = {1.908 ,1.908  ,1.908  ,1.908  ,1.908
   1.914 ,1.915  ,1.916  ,1.918  ,1.921  ,
   1.921 ,1.922  ,1.927  ,1.936  ,1.947  ,
   1.927 ,1.930  ,1.943  ,1.961  ,1.987  ,
   1.933 ,1.940  ,1.962  ,1.997  ,2.043  ,
   1.939 ,1.952  ,1.988  ,2.045  ,2.116  ,
   1.946 ,1.967  ,2.022  ,2.106  ,2.211  ,
   1.955 ,1.984  ,2.067  ,2.188  ,2.333  ,
   1.965 ,2.007  ,2.125  ,2.294  ,2.491  ,
   1.980 ,2.040  ,2.205  ,2.435  ,2.696  ,
   2.000 ,2.085  ,2.311  ,2.624  ,2.973  ,
   2.04  ,2.149  ,2.461  ,2.886  ,3.356  ,
   2.098 ,2.244  ,2.676  ,3.265  ,3.912  ,
   2.189 ,2.392  ,3.004  ,3.844  ,4.775  ,
   2.337 ,2.635  ,3.542  ,4.808  ,6.247  ,
   2.588 ,3.073  ,4.534  ,6.636  ,9.144  };
double nu_a[]= {2.0,1.9,1.8,1.7,1.6,1.5,1.4,1.3,1.2,
1.1,1.0,.9,.8,.7,.6,.5};
double nu_b[]= {0.0,0.25,0.5,0.75,1.0};
      i=0;
       while (i<16)
       {   if( x > nu_a[i])
       {   pt_x = i-1; break;
       }
       i++;
       }
      j=0;
       while (j<5)
       {
       if ( fabs(y) < nu_b[j])
       { pt_y=j-1;
       break;
         }
   j++;
       }
nc1 =interpolate(nu_a[pt_x],data[5*pt_x+pt_y],
  nu_a[pt_x+1],data[5*pt_x+pt_y+5],x);
```

```
     nc2 =interpolate(nu_a[pt_x],data[5*pt_x+pt_y+1],
      nu_a[pt_x+1],data[5*pt_x+pt_y+6],x);
nc3 =interpolate(nu_b[pt_y],nc1,nu_b[pt_y+1],nc2,fabs(y));
          return(nc3);}

double table_3(x,y)
double x,y;
{       int i,j,pta=15,ptb=4;
        double nd1,nd2,nd3;
 double dat[]={0.0 ,0.0    ,0.0      ,0.0      ,0.0    ,
        0.0 ,-.017  ,-.032    ,-.049    ,-.064   ,
        0.0 ,-.030  ,-.061    ,-.092    ,-.123   ,
        0.0 ,-.043  ,-.088    ,-.132    ,-.179   ,
        0.0 ,-.056  ,-.111    ,-.170    ,-.232   ,
        0.0 ,-.066  ,-.134    ,-.206    ,-.283   ,
        0.0 ,-.075  ,-.154    ,-.241    ,-.335   ,
        0.0 ,-.084  ,-.173    ,-.276    ,-.390   ,
        0.0 ,-.090  ,-.192    ,-.310    ,-.447   ,
        0.0 ,-.095  ,-.208    ,-.346    ,-.508   ,
        0.0 ,-.098  ,-.223    ,-.383    ,-.576   ,
        0.0 ,-.099  ,-.237    ,-.424    ,-.652   ,
        0.0 ,-.096  ,-.250    ,-.469    ,-.742   ,
        0.0 ,-.089  ,-.262    ,-.520    ,-.853   ,
        0.0 ,-.078  ,-.272    ,-.581    ,-.997   ,
        0.0 ,-.061  ,-.279    ,-.659    ,-1.198  };

double nua[]={2.0,1.9,1.8,1.7,1.6,1.5,1.4,1.3,1.2,1.1,1.0,
         .9,.8,.7,.6,.5};
double nub[]={0.0,0.25,0.50,0.75,1.00};

        i=0;
        while(i<16)
        {
        if( x > nua[i])
        { pta = i-1;
        break;
        }
        i++;
        }
        j=0;
        while(j<5)
        { if( fabs(y) < nub[j])
        { ptb =j-1; break; }
    j++;
         }
 nd1 = interpolate(nua[pta],dat[5*pta+ptb],
  nua[pta+1],dat[5*pta+ptb+5],x);
  nd2 = interpolate(nua[pta],dat[5*pta+ptb+1],
   nua[pta+1],dat[5*pta+ptb+6],x);
    nd3 = interpolate(nub[ptb],nd1,nub[ptb+1],nd2,fabs(y));
     nd3 = (y/fabs(y))*nd3;
   return(nd3);
}
```

stabfig.c

```
#include<stdio.h>
```

```
#include<math.h>
#include<stdlib.h>
#include<time.h>
#include"supdir.h"
#include"supdir.c"

main(int argc,char *argv[])
{    FILE *datfile,*outfile1,*outfile2,*outfile3;
   double *stack,*stock,*samp[2];
   double *param,x,val;
   double temp,left,rt,h,xx;
   int i,j,k,start,stop,count,cnt,len,range,*sampdim;
   int x_lim;
   char *data;
double rule1(double *samp[2],double x,double *par);
double rule2(double *samp[2],double x,double *par);
void up_order(double *samp,int sampdim);
double qtile(double *samp,double x,int sampdim);

  data = argv[1];
  if (data == NULL)
  {
     printf("combin [datafile] [outfile]\n");
  exit(0);
     }
   datfile = fopen(data,"r");
  if (datfile == NULL)
  {
  printf("No such file exists. Try again.\n");
  exit(0);
     }

  count=0;
  while( fscanf(datfile,"%lf",&temp) != EOF)
  {count++;}
  rewind(datfile);
  stock = v_alloc(count);
  for(i=0;i<count;i++)
  { fscanf(datfile,"%lf",&temp);
     *(stock+i) = log(temp);
      }
 printf("\nTotal no. of data points in file = %d\n",count);

 stack=v_alloc(count-1);
for(i=0;i<count-1;i++)
 *(stack+i)=*(stock+i+1)-(*(stock+i));
 free(stock);

 sampdim=v_ialloc(2);
for(i=0;i<2;i++){
printf("\nType %d-st sample starting lower limit\n",i+1);
scanf("%d",&start);
printf("\nType %d-st sample end, upper limit\n",i+1);
scanf("%d",&stop);
len=stop-start+1;
*(sampdim+i)=len;
samp[i]=v_alloc(len);
```

```
for(j=0;j<len;j++) *(samp[i]+j)=*(stack+start-1+j);
}
fclose(datfile);
up_order(samp[0],*sampdim);
up_order(samp[1],*(sampdim+1));
for(i=0;i<*sampdim;i++)
printf("\nsamp[0] +%d =%lf\n",i,*(samp[0]+i));

left= *samp[0] > *samp[1] ? *samp[0] : *samp[1];
rt= *(samp[0]+(*sampdim)-1) < *(samp[1]+(*(sampdim+1))-1) ?
*(samp[0]+(*sampdim)-1) : *(samp[1]+(*(sampdim+1))-1);
printf("\nleft = %lf, rt = %lf\n",left,rt);

param=v_alloc(2);
*param = (double) *sampdim;
*(param+1)=(double) *(sampdim+1);
x_lim=50;
h=(rt-left)/((double)x_lim);
xx=left;
outfile1=fopen("stplot1.dat","w");
for(i=0;i<x_lim;i++)
{
fprintf(outfile1,"%lf %lf %lf\n",xx,rule1(samp,xx,param),
 rule2(samp,xx,param));
xx+=h;
}
outfile2=fopen("stplot2.dat","w");
for(i=0;i<*sampdim-1;i++)
{
xx=*(samp[0]+i);
fprintf(outfile2,"%lf %lf\n",xx,-log(qtile(samp[0],xx,*sampdim)));
}

outfile3=fopen("stplot3.dat","w");
for(i=0;i<*(sampdim+1)-1;i++)
{
xx=*(samp[1]+i);
fprintf(outfile3,"%lf %lf\n",xx,-log(qtile(samp[1],xx,*(sampdim+1))));
}}

double rule1(double *samp[2],double x,double *par)
{
int i,n1,n2;
double val,np1,np2,n;
n1=(int) (*par);
n2=(int) (*(par+1));
n= (*(par+1))+(*par);
np1=np2=0.0;
for(i=0;i<n1;i++)
np1 += *(samp[0]+i) >x ? 1.0 : 0.0;
for(i=0;i<n2;i++)
np2 += *(samp[1]+i) >x ? 1.0 : 0.0;
np1 /=n;
np2 /=n;
val=fabs(np2-0.5)-fabs(np1-0.5);
return(val);
}
```

```
double rule2(double *samp[2],double x,double *par)
{
int i,n1,n2;
double val,np1,np2,n,pi1,pi2;

n1=(int) (*par);
n2=(int) (*(par+1));
n= (*(par+1))+(*par);
pi1= *par/n;
pi2=*(par+1)/n;

np1=np2=0.0;
for(i=0;i<n1;i++)
np1 += *(samp[0]+i) >x ? 1.0 : 0.0;
for(i=0;i<n2;i++)
np2 += *(samp[1]+i) >x ? 1.0 : 0.0;
np1 /=n;
np2 /=n;
val=(pi2*fabs(np2-0.5))-(pi1*fabs(np1-0.5));
return(val);
}

double qtile(double *samp,double x,int sampdim)
{
int i;
double val=0.0;
for(i=0;i<sampdim;i++)
val += *(samp+i) >x ? 1.0 : 0.0;
val /=(double)sampdim;
return(val);
}
void up_order(double *samp,int sampdim)
{
int i,j,k;
double temp;
for(i=0;i<sampdim;i++)
{for(j=i+1;j<sampdim;j++)
 {if(*(samp+j) < *(samp+i)){temp=*(samp+j);
       *(samp+j)=*(samp+i);
       *(samp+i)=temp;
      }}}}
```

stab1.inp

```
set terminal latex
set output "stabr1.tex"
set size 4.5/5.,3.8/3.
#set format xy "$%g$
set title "Plot two rules"
set xlabel "$x$" 0,-1
set ylabel "$rule value$" -6
set nokey
set label "rule1" at 0.08,0.12
set label "rule2" at 0.08,0.1
set xtics  -0.14,0.04
plot "stplot1.dat" using 1:2 w l,"stplot1.dat" using 1:3 w l
```

stab2.inp

```
set terminal latex
set output "stabr2.tex"
set size 4.5/5.,3.5/3.
#set format xy "$%g$
set title "Plot of -log(quantile) for two samples"
set xlabel "$x$" 0,-2
set ylabel "$-log(quantile)$" -7
set nokey
set label "1st sample (1-320)" at 0.18,6 left
set label "2nd sample (321-614)" at 0.18,5.5  left
#set xtics
plot "stplot2.dat" w l,"stplot3.dat" w l
```

REFERENCES

1. V.M. Zolotarev, *One Dimensional Stable Distributions*, AMS Translation of Mathematical Monographs, Providence, RI, (1986).
2. S. Mittnik, S.T. Rachev and M.S. Paolella, Modeling asset returns with alternative stable distributions, *Econometric Reviews* **12**, 261–330, (1993).
3. S.T. Rachev, A program for computing the estimates of a 4-parameter stable distribution, Private Communication, (1997).
4. G. Samorodnitsky and M.S. Taqqu, *Stable Non-Gaussian Random Processes*, Chapman and Hall, New York, (1994).
5. A. SenGupta and S. Roy, On consistent and efficient estimation of Cox's logistic discrimination rule, (submitted).
6. A. SenGupta and S. Roy, On families admitting logistic discrimination rule: Characterizing the multivariate, multiparameter case, Under revision, (2001).