LA-UR-19-31480
Approved for public release; distribution is unlimited.

Title: Quantification of Margins and Uncertainty for Multicomponent Systems

Author(s): Marcy, Peter William
Williams, Brian J.
Tippetts, Trevor Bair

Intended for: Report

Issued: 2019-11-14

Quantification of Margins and Uncertainty for Multicomponent Systems

Peter W. Marcy, Brian J. Williams, Trevor B. Tippetts

November 14, 2019

1 Introduction

This report describes and illustrates several metrics for application to the analysis of a multicomponent system. Specifically, three metrics are calculated for each system component. The first metric is component failure probability (π). This will typically be expressed as the probability that a component performance variable (P) exceeds a specified or random threshold (T). The *margin* (M) for component performance is defined as $M = P - T$ and will be the focus of analysis throughout this report. The second metric is referred to as *margin to failure rate* ϕ (MTF_ϕ). This is an estimate of the allowable "slack" in component performance that could be tolerated to modify its failure rate from π to a user-specified value ϕ. The third and final metric is *margin sensitivity* (MS). This quantity orders components based on the degree to which changes in their failure rates are affected by shifts in their margin distributions. We first motivate the metrics in an idealized setting, and then expand to the typical application scenarios where they must be estimated from experimental or simulated data. In this environment, we discuss methods for estimating the metrics *with uncertainty quantification.*

2 The Metrics

Suppose that there is a multicomponent system and that each of the S components (subsystems) has failure probability π_s ($s = 1, \ldots, S$) which depends on a number of factors. The whole system functions via serial and parallel operations of subsystems and so the joint failure probability $\pi^{\text{sys}} = \mathbb{P}(\text{system failure})$ may be computed as follows. If it is an entirely parallel system and all components are statistically independent of one another, then

$$\pi^{\text{sys}}_{\text{par}} = \mathbb{P}(\text{all components fail}) = \prod_{s=1}^{S} \pi_s \,. \tag{1}$$

If, on the other hand, it is an entirely serial system, then

$$\pi^{\text{sys}}_{\text{ser}} = 1 - \mathbb{P}(\text{all components work}) = 1 - \prod_{s=1}^{S} (1 - \pi_s) \,. \tag{2}$$

For mixture systems these rules can be combined to compute the overall π^{sys}. The full system reliability is then $1 - \pi^{\text{sys}}$. The **first** and most important set of metrics for analyzing system failure or reliability is then $\{\pi_s\}_{s=1}^{S}$. We will motivate and provide two more metrics below.

Suppose further that there is a latent mechanism such that a subsystem fails when some physical margin is (without loss of generality) positive; that is, $\pi_s = \mathbb{P}(M_s > 0)$. This quantity can also be

written as $1 - F_{M_s}(m = 0) = \int_{m=0}^{\infty} f_{M_s}(u)du$, where F_{M_s} is the cumulative distribution function (cdf) and f_{M_s} is probability density function (pdf) for the random variable M_s.

If a margin variable is shifted by m_s, i.e. $M_s \to (M_s + m_s)$, the resulting failure probability is the function

$$\pi_s(m_s) = \mathbb{P}\left(M_s + m_s > 0\right) = \mathbb{P}\left(M_s > -m_s\right) = 1 - F_{M_s}(-m_s)$$

(the original failure rates are $\pi_s \equiv \pi_s(0)$ for all s). Substituting this into (1) and (2) gives the functions

$$\pi_{\text{par}}^{\text{sys}}(m_1, \ldots, m_S) = \prod_{s=1}^{S}[1 - F_{M_s}(-m_s)] \qquad \text{and} \qquad \pi_{\text{ser}}^{\text{sys}}(m_1, \ldots, m_S) = 1 - \prod_{s=1}^{S} F_{M_s}(-m_s)\,. \quad (3)$$

Using the above expressions (combining, if necessary), an analyst may then see the trade-offs in the overall system failure probability as a function of shifts in the individual margin variables.

The analyst may ask what shift of a particular margin variable gives a prescribed failure rate of ϕ_s. A **second** metric is the answer to this question; the margin to failure rate ϕ_s (MTF_{ϕ_s}) is negative of the $(1-\phi_s)^{\text{th}}$ quantile of M_s since by definition, $Q_{1-\phi_s}$ satisfies $1 - F_{M_s}(Q_{1-\phi_s}) = \phi_s$. Note that the analyst can also use the quantile to investigate a simultaneous shift (by m_s) and scale (by σ_s) to produce a change in the failure probability. If a success rate of 99% were desired then

$$0.99 \overset{\text{set}}{=} \mathbb{P}\left(\sigma_s M + m_s \leq 0\right) = \mathbb{P}\left(M \leq \frac{-m_s}{\sigma_s}\right) = F_M(-m_s/\sigma_s)$$
$$\Rightarrow -Q_{1-0.01} \overset{\text{set}}{=} \frac{m_s}{\sigma_s}\,.$$

A **third** metric is related and comes from taking differentials

$$d\pi_s(m_s) = d\left(1 - F_{M_s}(-m_s)\right) = \frac{-dF_{M_s}}{dm_s} \cdot -dm_s = f_{M_s}(m_s) \cdot dm_s\,,$$

so that

$$\begin{aligned}\Delta\pi_s = (\phi_s - \pi_s) &\approx f_{M_s}(0) \cdot (\Delta m_s = -Q_{1-\phi_s} - 0) \\ &= f_{M_s}(0) \cdot (-Q_{1-\phi_s})\,.\end{aligned}$$

This third metric, margin sensitivity $\text{MS}_s = f_{M_s}(0)$, is a relative measure of sensitivity to small changes in the component margin distributions. The rearrangement of $f_{M_1}(0), \ldots, f_{M_S}(0)$ into descending order provides a ranking of the importance of the variables $M_1, \ldots, M_S$ to the system reliability (when perturbed by the same $\Delta m_1 = \ldots = \Delta m_S$).

The three (sets of) metrics we have proposed for multicomponent failure analysis appear when linearizing the system failure probability. Differentials for parallel and serial systems are, as a function of the shift variables,

$$\begin{aligned} d\left(\pi_{\text{par}}^{\text{sys}}\right) = d\prod_{s=1}^{S}\pi_s(m_s) \qquad &= \left(\prod_{s=1}^{S}\pi_s(m_s)\right) \cdot \left(\sum_{s=1}^{S}\frac{1}{\pi_s(m_s)}\, d\pi_s(m_s)\right) \\ d\left(\pi_{\text{ser}}^{\text{sys}}\right) = d\Big(1 - \prod_{s=1}^{S}[1 - \pi_s(m_s)]\Big) \quad &= -\left(\prod_{s=1}^{S}[1 - \pi_s(m_s)]\right) \cdot \left(\sum_{s=1}^{S}\frac{1}{1 - \pi_s(m_s)}\, (-d\pi_s(m_s))\right). \end{aligned}$$

Thus at $m_1 = \ldots = m_S = 0$ these differentials imply that

$$\Delta\left(\pi_{\text{par}}^{\text{sys}}\right) \;\approx\; d\left(\pi_{\text{par}}^{\text{sys}}\right) = \pi_{\text{par}}^{\text{sys}} \cdot \sum_{s=1}^{S} \frac{f_{M_s}(0)}{\pi_s} \cdot (\Delta m_s = -Q_{1-\phi_s})$$

$$\Delta\left(\pi_{\text{ser}}^{\text{sys}}\right) \;\approx\; d\left(\pi_{\text{ser}}^{\text{sys}}\right) = \pi_{\text{ser}}^{\text{sys}} \cdot \sum_{s=1}^{S} \frac{f_{M_s}(0)}{1-\pi_s} \cdot (\Delta m_s = -Q_{1-\phi_s}) \;,$$

so that in both cases feature $\{\pi_s\}$, $\{\text{MTF}_{\phi_s} = -Q_{1-\phi_s}\}$, and $\{\text{MS}_s = f_{M_s}(0)\}$, as desired. This is purely illustrative, for one wishing to actually compute the change in total probability would instead use (3) with $m_s = -Q_{1-\phi_s}$.

The discussion above provides motivation for three metrics, and now to solidify the concepts, consider the following analytic (closed-form, exact) examples in hypothetical two-component ($S = 2$) systems.

2.1 Example: Two Normal Margins in Parallel

Suppose that two subsystems operate in parallel with Normal margin distributions

$$M_1 \sim N(\mu_1 = -2, \sigma_1 = 1.0)$$
$$M_2 \sim N(\mu_2 = -3, \sigma_2 = 1.6)\,.$$

Let $\Phi(\cdot)$ denote the standard normal cdf function. Then the failure probability for each system (first metric) is $\pi_1 = 1 - \Phi(\frac{0+2}{1.0}) = 0.0228$ and $\pi_2 = 1 - \Phi(\frac{0+3}{1.6}) = 0.0304$; the system failure probability is the product of these two numbers: $\pi_{\text{par}}^{\text{sys}} = 0.0007$.

For the second metric, to achieve a $\phi = 0.01$ (for example) failure probability in each margin, each would have to be shifted by $\text{MTF}_{\phi_1} = -Q_{N,0.99}^{1} = -F_N^{-1}(0.99; \mu_1, \sigma_1) = -0.3263$ and $\text{MTF}_{\phi_2} = -Q_{N,0.99}^{2} = -F_N^{-1}(0.99; \mu_2, \sigma_2) = -0.7222$ units. Note that both are negative values meaning a shift to the left, as expected.

And as for the third metric, the sensitivities are the respective normal densities evaluated at the origin; these are $\text{MS}_1 = 0.0540$ and $\text{MS}_2 = 0.0430$. This would imply that the "easiest" way to change the system failure probability would be to change the failure probability for the first component. ♦

2.2 Example: Two Generalized Pareto Margins in Series

Now suppose that two subsystems operate in series with generalized Pareto distributions (GPDs) as margins,

$$M_1 \sim GPD(\mu_1 = -10, \sigma_1 = 1.0, \xi_1 = 1)$$
$$M_2 \sim GPD(\mu_2 = -25, \sigma_2 = 0.5, \xi_2 = 1.5)\,.$$

The GPD density and cumulative distribution functions are, respectively

$$f_{GPD}(x; \mu, \sigma, \xi) = \frac{1}{\sigma}\Big(1 + \frac{\xi(x-\mu)}{\sigma}\Big)^{-\frac{1}{\xi}-1} \tag{4}$$

$$F_{GPD}(x; \mu, \sigma, \xi) = 1 - \Big(1 + \frac{\xi(x-\mu)}{\sigma}\Big)^{-\frac{1}{\xi}}\,, \tag{5}$$

either for $x \geq \mu$ and $\xi \geq 0$; or for $\mu \leq x \leq \mu - \sigma/\xi$ and $\xi < 0$. When $\xi = 0$ the GPD becomes a shifted and scaled exponential distribution.

The failure probability for each system is derived from $\pi_s = 1 - F_{GPD}(0; \mu_s, \sigma_s, \xi_s)$ for $s = 1, 2$; specifically these values are $\pi_1 = 0.09091$ and $\pi_2 = 0.05573$. The system failure probability is $\pi_{\text{ser}}^{\text{sys}} = 1 - (1 - 0.09091) \times (1 - 0.05573) = 0.14158$.

For the second metric, to achieve a $\phi = 0.01$ (again, just for illustration) failure probability in each margin, each would have to be shifted by $\text{MTF}_{\phi_1} = -Q^1_{GPD,0.99} = -F^{-1}_{GPD}(0.99; \mu_1, \sigma_1, \xi_1) = -89$ and $\text{MTF}_{\phi_2} = -Q^2_{GPD,0.99} = -F^{-1}_{GPD}(0.99; \mu_2, \sigma_2, \xi_2) = -308$ units. These seemingly extreme numbers are necessary due to the heavy tailed nature of the GPD.

And as for the third metric, the sensitivities are the respective densities above evaluated at the origin; these are $\text{MS}_1 = 0.00826$ and $\text{MS}_2 = 0.00147$. Again it is the case that the system failure probability is more sensitive to the first component. ♦

Closed-form examples are tidy and useful, but in reality each metric must be estimated from data. Furthermore, the resulting uncertainties in $\{\hat{\pi}_s\}$, $\{-\hat{Q}_{1-\phi_s}\}$, and $\{\hat{f}_{M_s}(0)\}$ (especially the first) can also be used by a systems analyst. For example, an analyst should be interested not only in components with large failure probabilities but also those with considerable uncertainty in the estimates. The purpose of the remainder of this report is to detail the inference, i.e. estimation and uncertainty quantification, for the three metrics when they are obtained from data. This involves two main cases. In the first, a margin random variable M is measured directly, as in the discussion above. In the second, there is a performance variable P and a threshold variable T; a failure occurs when the performance exceeds the threshold, $M = P - T > 0$. Data is obtained on both of these random variables and the three metrics are to be determined from both samples.

For the sake of expositional clarity, note that *in the remainder of the report we will focus on inference involving a single margin variable, so that "s" subscripts can be dropped; instead a subscript will refer to an observation number.*

3 When the Margin Variable is Observed Directly

It is worth noting that a margin will often depend on controllable factors $\boldsymbol{x}$ and unknown physical parameters $\boldsymbol{\theta}$ (i.e., $M \equiv M(\boldsymbol{x}, \boldsymbol{\theta})$), but we will ignore this dependence for the sake of clarity. However, all of what we present could be expanded to a more detailed discussion in which knowledge of $\boldsymbol{x}$ and $\boldsymbol{\theta}$ could be used in an analyst's uncertainty analysis.

When independent and identically distributed (iid) samples $M_1, \ldots, M_N$ are observed, the estimators are straightforward. In what follows $\mathbb{I}\{\cdot\}$ is an indicator function and $M_{(1)} < \cdots < M_{(N)}$ are the order statistics; also assume that there are F observed failures (F of the N cases have

$M > 0$):

$$\hat{\pi} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}\{M_n > 0\} = \frac{F}{N} \qquad \text{sample proportion of positive margins} \quad (6)$$

$$\hat{Q}_{1-\phi} \text{ such that } \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}\left\{M_i \le \hat{Q}_{1-\phi}\right\} \approx 1-\phi \qquad \text{sample } (1-\phi) \text{ quantile} \quad (7)$$

$$= M_{(\lfloor v \rfloor)} + (v - \lfloor v \rfloor)(M_{(\lfloor v \rfloor + 1)} - M_{(\lfloor v \rfloor)}) \qquad v = (N + 1/3)(1-\phi) + 1/3$$

$$\hat{S} = \hat{f}_M(0; h) = \frac{1}{Nh}\sum_{n=1}^{N} K\left(\frac{M_n}{h}\right) \qquad \text{kernel density estimate at } 0\,. \quad (8)$$

The first metric (estimated failure probability) is straightforward and needs no further explanation. The second metric (MTF_ϕ) is one of many definitions of the appropriate sample quantile. In the case of the third metric (MS), the bandwidth h must be selected. Usually this is chosen to satisfy a goodness-of-fit criterion across the entire range of the data ([16], [18]). We note that because we only care about one point ($m = 0$), which is likely in or close to the right tail, we could appeal to a more tailored criterion (i.e. mean squared error at $m = 0$ instead of *integrated* mean squared error). There are other more sophisticated options, such as those based on transformations ([17], [3]) which are worth mentioning, but we do not discuss here.

Uncertainty in each of these metrics are reflected in confidence intervals. These can be obtained through *parametric* techniques wherein a distributional family is assumed for $M_1, \ldots, M_N$, parameters are estimated, and the metrics are derived using equations above. We shall however discuss classical, *nonparametric* intervals wherein no distributional assumptions are required. More sophisticated machinery is required when $M_1, \ldots, M_N$ are samples from a heavy-tailed distribution; this is covered in Section 3.1. This same machinery is also useful in the case that F (number of observed failures) or ϕ (prescribed failure rate) is small.

A conservative $100(1-\alpha)\%$ interval for the first metric is obtained by a direct method known as "pivoting the cdf" (see [12], pgs. 103, 465) and is given by

$$[\underline{\pi}, \widetilde{\pi}] = \left[\texttt{qbeta}\left(\frac{\alpha}{2};\ F,\ N-F+1\right),\ \texttt{qbeta}\left(1-\frac{\alpha}{2};\ F+1,\ N-F\right)\right]$$

where "$\texttt{qbeta}(p;\ a, b)$" is the p^{th} quantile of the Beta distribution having parameters a, b. An improvement comes through a slight modification known as Jeffreys Approximate Method ([12], pg. 107); this interval takes the form

$$[\underline{\pi}, \widetilde{\pi}] = \left[\texttt{qbeta}\left(\frac{\alpha}{2};\ F+0.5,\ N-F+0.5\right),\ \texttt{qbeta}\left(1-\frac{\alpha}{2};\ F+0.5,\ N-F+0.5\right)\right]. \quad (9)$$

A conservative $100(1-\alpha)\%$ confidence interval for $Q_{1-\phi}$ can also be obtained by pivoting the cdf of the order statistics ([12], pg. 498-499). For the given level of coverage, integers $1 \le l < u \le N$ are sought such that the interval $[M_{(l)}, M_{(u)}]$ satisfies

$$\begin{aligned}
1-\alpha &\overset{\text{set}}{\le} \mathbb{P}\left(M_{(l)} \le Q_{1-\phi} \le M_{(u)}\right) = \mathbb{P}\left(M_{(l)} \le Q_{1-\phi}\right) - \mathbb{P}\left(M_{(u)} \le Q_{1-\phi}\right) \\
&= \mathbb{P}\left(F_M(M_{(l)}) \le 1-\phi\right) - \mathbb{P}\left(F_M(M_{(u)}) \le 1-\phi\right) \\
&= \mathbb{P}\left(U_{(l)} \le 1-\phi\right) - \mathbb{P}\left(U_{(u)} \le 1-\phi\right) \\
&= \texttt{pbeta}\,(1-\phi;\ l,\ N-l+1)\ -\ \texttt{pbeta}\,(1-\phi;\ u,\ N-u+1) \\
&= \texttt{pbinom}\,(u-1;\ N,\ 1-\phi)\ -\ \texttt{pbinom}\,(l-1;\ N,\ 1-\phi)
\end{aligned}$$

with maximal l and minimal u. Above it was used that if $U_{(l)}$ is the lth order statistic in a size N sample of $Unif(0,1)$ random variables then $U_{(l)} \sim Beta(l, N-l+1)$; "`pbeta`" and "`pbinom`" denote the cdf of the beta and binomial distributions. In addition, the binomial and beta cdfs are related by

$$F_{Bin}(x;\ N, p) = 1 - F_{Beta}(p;\ x+1, N-x)$$

(see [12] (C.19) pg. 439); this identity is most useful when N is large. As for how to choose l and u, one may proceed by finding two one-sided $100(1-\alpha/2)\%$ intervals; that is, given the observed order statistics, find l and u such that

$$\begin{aligned}
l &\stackrel{\text{def}}{=} \arg\max_r \mathbb{P}\left(M_{(r)} \le Q_{1-\phi}\right) \ge 1-\alpha/2 \\
&\equiv \arg\max_r 1 - \texttt{pbinom}\,(r-1;\ N,\ 1-\phi) \ge 1-\alpha/2 \\
&\equiv \arg\max_r \texttt{pbinom}\,(r-1;\ N,\ 1-\phi) \le \alpha/2 \\
&= \texttt{qbinom}\,(\alpha/2;\ N,\ 1-\phi) \\
u &\stackrel{\text{def}}{=} \arg\min_r \mathbb{P}\left(M_{(r)} \ge Q_{1-\phi}\right) \ge 1-\alpha/2 \\
&\equiv \arg\min_r \texttt{pbinom}\,(r-1;\ N,\ 1-\phi) \ge 1-\alpha/2 \\
&= \texttt{qbinom}\,(1-\alpha/2;\ N,\ 1-\phi) + 1\,.
\end{aligned}$$

Improvements are possible by linearly interpolating order statistics ([13], [2]). That is, an adjusted interval can be formed using adjacent order statistics,

$$\begin{aligned}
&\mathbb{P}\big(Q_{1-\phi} < (1-\lambda_l)M_{(l)} + \lambda_l M_{(l+1)}\big) \approx \alpha/2 \\
&\mathbb{P}\big(Q_{1-\phi} < (1-\lambda_u)M_{(u)} + \lambda_u M_{(u+1)}\big) \approx 1-\alpha/2 \\
\Rightarrow \quad &[\underset{\sim}{Q}_{1-\phi}, \widetilde{Q}_{1-\phi}] = \big[(1-\lambda_l)M_{(l)} + \lambda_l M_{(l+1)}, (1-\lambda_u)M_{(u)} + \lambda_u M_{(u+1)}\big] \qquad (10)
\end{aligned}$$

after solving for the weights λ_l, λ_u in the final equation of [13] (pg. 131).

Finally, on to the third metric. Typically in kernel density estimation, a bandwidth is chosen to *undersmooth* the data in order to get an unbiased, though higher variance, estimator of the density. Then using a plug-in or bootstrap estimate of the variance allows for a confidence interval with near nominal coverage. The following variance estimate can be used to form an approximate $100(1-\alpha)\%$ interval ([8], bottom of pg. 678):

$$\begin{aligned}
s_N^2 &\stackrel{\text{def}}{=} \widehat{Var}(\hat{S}) = \frac{1}{(Nh)^2}\sum_{n=1}^{N} K^2\left(\frac{M_n}{h}\right) - \frac{\hat{S}^2}{N} \\
\Rightarrow \quad [\underset{\sim}{S}, \widetilde{S}] &= \Big[\hat{S} + \texttt{qnorm}(\alpha/2)s_N,\ \hat{S} + \texttt{qnorm}(1-\alpha/2)s_N\Big]\,. \qquad (11)
\end{aligned}$$

`R` code to obtain the metrics as well as perform a simulation study for confidence interval coverage is given in Appendix A.1. We end by noting that while it may be tempting to obtain uncertainties in the metrics via simple bootstrap resampling [5], this is not advisable. There are theoretical reasons, but the bottom line is that intervals found in this manner do not have proper frequentist coverage. Taking the third metric as an example, [5] (pg. 226) reports that, "bootstrap confidence intervals for the value of a density raise some awkward issues." (The authors then go on to motivate the need for transformations, a double bootstrap, and careful choice of the bandwidth.) By comparison, in our investigations, the simple nonparametric and asymptotic intervals provided earlier in this section outperformed the bootstrap intervals and required less work.

3.0.1 Section 2 Examples (continued)

Here we demonstrate the potential for UQ by constructing confidence intervals using the methods above, within the context of the two examples of Sections 2.1 and 2.2. Samples of size 1000 are drawn for two margins M_1 and M_2 when these have either normal or generalized Pareto (GPD) distributions with parameters given in Sections 2.1 and 2.2. Point and interval estimates are given in Table 1. The estimated values are derived from equations (6) – (8); the upper and lower bounds for the intervals are derived from equations (9) – (11). `R` code to produce the numbers within the table is given in Appendix A.1.

For this one particular realization, the confidence intervals cover the true values in all but one of the cases: the third metric for the second GPD margin. In a further simulation study (code also provided in Appendix A.1), these trends held — intervals for MS always had lower than nominal coverage compared to the other two metrics which behaved favorably. ♦

		M_1			M_2		
Example	Metric	Lower	Estimate	Upper	Lower	Estimate	Upper
Normal	π	0.0111	0.0180	0.0277	0.0166	0.0250	0.0361
	MTF_ϕ	-0.501	-0.2405	-0.0286	-1.2745	-0.8562	-0.2709
	MS	0.0538	0.0703	0.0869	0.0292	0.0386	0.0480
GPD	π	0.0799	0.0970	0.1165	0.0457	0.0590	0.0749
	MTF_ϕ	-200.72	-97.004	-46.423	-591.65	-266.04	-82.232
	MS	0.0049	0.0091	0.0134	-4.44×10^{-6}	8.12×10^{-5}	0.00016

Table 1: Estimates together with lower and upper bounds of 95% confidence intervals for the true values of the metrics $(\pi, \text{MTF}_\phi, \text{MS})$. Intervals failing to cover the true value are highlighted in red.

3.1 When M Has Heavy Tail

It is possible that the margin distribution F_M is not just skewed, but heavy-tailed with extreme tail index ξ, in which case more machinery is needed. To motivate computation of the three metrics in this setting, we begin with a summary of relevant concepts from extreme value theory. A function $\ell(x)$ is said to be *slowly varying* if for any $t > 0$,

$$\lim_{x\to\infty} \frac{\ell(tx)}{\ell(x)} = 1\,.$$

The margin distribution F_M is *heavy-tailed* if

$$F_M(x) = 1 - x^{-1/\xi}\,\ell(x)\,,\; x > 0\,,$$

for some slowly varying function ℓ and $\xi > 0$.

Given threshold $u_m > 0$, the *excess distribution function* F_{M,u_m} for the margin M is defined by

$$F_{M,u_m}(x) = \mathbb{P}[\,M - u_m \le x \,|\, M > u_m\,] = \frac{F_M(x+u_m) - F_M(u_m)}{1 - F_M(u_m)}\,.$$

Recalling definition (5) of the GPD, the *peaks over threshold* (POT) theorem ([1], [14]) states that F_M is heavy-tailed with index $\xi > 0$ if and only if there exists $\sigma(u_m)$ such that

$$\lim_{u_m\to\infty} \sup_x |F_{M,u_m}(x) - F_{GPD}(x; 0, \sigma(u_m), \xi)| = 0\,.$$

It is shown in [11] that in the context of the POT theorem, $\sigma(u_m)$ is asymptotically ($u_m \to \infty$) equivalent to ξu_m, in which case the GPD becomes a shifted Pareto distribution,

$$F_{\hat{P}}(x; u_m, \xi) = 1 - \left(\frac{u_m}{x + u_m}\right)^{1/\xi} \text{ for } x > 0\,.$$

The practical import of the POT theorem is that the tail of the unknown margin distribution F_M can be approximated in two ways for the purpose of estimating the metrics. The first utilizes a GPD(u_m, σ, ξ) distribution,

$$\hat{F}_M(x) = p_{u_m} + (1 - p_{u_m}) F_{GPD}(x; u_m, \sigma, \xi) \text{ for } x > u_m\,, \tag{12}$$

where $p_{u_m} = \mathbb{P}[\, M \le u_m \,]$ and $F_{GPD}(\cdot; u_m, \sigma, \xi)$ is given by equation (5). The second utilizes a Pareto(u_m, ξ) distribution,

$$\hat{F}_M(x) = p_{u_m} + (1 - p_{u_m}) F_P(x; u_m, \xi) \text{ for } x > u_m\,, \tag{13}$$

where

$$F_P(x; u_m, \xi) = 1 - \left(\frac{u_m}{x}\right)^{1/\xi} \text{ for } x > u_m\,.$$

For the benefit of the subsequent discussion, observe that the Pareto approximation (13) to the margin tail is obtained by setting $\sigma = \xi u_m$ in the GPD approximation (12).

We assume without loss of generality that M has been shifted by a specified constant c to ensure $u_m > 0$. Although such a shift is easily accommodated theoretically for metric calculation, it has a potentially non-negligible practical impact on tail estimation. Simulation studies suggest that such shifts should be limited as much as possible. For example, removing observations clearly not part of the relevant tail (say, keeping 15% – 40% of the observations in the tail of interest) prior to tail estimation has worked well in practice. See Section 3.1.2 for additional discussion.

The first metric (failure probability π) is approximated as follows,

$$\pi = \mathbb{P}[\, M > c \,] = 1 - \hat{F}_M(c) = (1 - p_{u_m})(1 - F_{GPD}(c; u_m, \sigma, \xi)) \text{ for } c > u_m\,.$$

The second metric (margin to failure rate ϕ, MTF_ϕ) solves the equation $\phi = \mathbb{P}[\, M + \text{MTF}_\phi > c \,]$ and is approximated by

$$\text{MTF}_\phi = c - Q_{GPD,(1-p_{u_m}-\phi)/(1-p_{u_m})} \text{ for } p_{u_m} + \phi < 1\,.$$

Here $Q_{GPD,\alpha} = u_m + (\sigma/\xi)[(1-\alpha)^{-\xi} - 1]$ is the α quantile of the GPD(u_m, σ, ξ) distribution, giving

$$\text{MTF}_\phi = c - u_m - \frac{\sigma}{\xi}\left[\left(\frac{1 - p_{u_m}}{\phi}\right)^{\xi} - 1\right] \text{ for } p_{u_m} + \phi < 1\,.$$

The third metric (margin sensitivity, MS) is approximated by

$$\text{MS} = (1 - p_{u_m}) f_{GPD}(c; u_m, \sigma, \xi) \text{ for } c > u_m\,,$$

where $f_{GPD}(\cdot; u_m, \sigma, \xi)$ is given by equation (4).

Use of the GPD approximation implies the ability to estimate the threshold, scale, and tail index triplet (u_m, σ, ξ), say by $(\hat{u}_m, \hat{\sigma}, \hat{\xi})$. This simplifies in the case of the Pareto approximation, reducing

to the estimation of (u_m, ξ) by $(\hat{u}_m, \hat{\xi})$. Given samples $M_1, M_2, \ldots, M_N$ from the margin distribution F_M, shifted by $c = -M_{(1)}+\varepsilon$ (for any $\varepsilon > 0$) if necessary (see caveat above) to guarantee positive tail observations (if not, set $c = 0$), form the corresponding order statistics $M_{(1)} < M_{(2)} < \cdots < M_{(N)}$. The basic idea is to identify the set of observations $M_{(N-k)}, M_{(N-k+1)}, \ldots, M_{(N)}$ constituting the tail via choice of integer k. The tail index (and scale if necessary) are then estimated from this subset of observations. We utilize the double bootstrap method of [4] as improved by [15] to estimate k. The estimate $\hat{k}$ of k is found by minimizing the estimated asymptotic mean squared error of the Hill estimator [9] of ξ given by

$$\hat{\xi}_h = \frac{1}{k}\sum_{i=1}^{k} \log M_{(N-i+1)} - \log M_{(N-k)}\,. \tag{14}$$

The threshold is estimated by $\hat{u}_m = M_{(N-\hat{k})}$. Three approaches to estimating the remaining parameter(s) using the tail observations $M_{(N-\hat{k})}, M_{(N-\hat{k}+1)}, \ldots, M_{(N)}$ are considered:

- GPD: Use the GPD$(\hat{u}_m, \sigma, \xi)$ tail approximation and set $(\hat{\sigma}, \hat{\xi})$ to the maximum likelihood estimates of (σ, ξ). These can be found using the `fpot` function in the `R` package `evd`.
- Pareto-I: Use the Pareto$(\hat{u}_m, \xi)$ tail approximation and set $\hat{\xi}$ to the maximum likelihood estimate of ξ, which is straightforwardly shown to be the Hill estimate $\hat{\xi}_h$ (14).
- Pareto-II: Use the Pareto$(\hat{u}_m, \xi)$ tail approximation. Fit a linear regression to the I/O pairs $(\log M_{(N-\hat{k}+i)}, \log(\hat{k} - i + 1/2))$, $i = 1, \ldots, \hat{k}$, and set $\hat{\xi} = -1/\hat{b}$ where $\hat{b}$ is the least squares estimate of the slope (see [7]).

If $c > \hat{u}_m$ and $\phi < \hat{k}/N$, the metrics above can be estimated by taking $p_{u_m} = (N - \hat{k})/N$ and using estimates of the remaining parameters calculated as just described.

Uncertainty quantification for each of the three metrics and estimation approaches is based on the Delta method ([6], Theorem 7), *conditional* on the value of $\hat{u}_m$ found from the double bootstrap. To obtain confidence intervals for each metric, transformations $(Q_1, Q_2, Q_3) \leftarrow (\pi, \mathrm{MTF}_\phi, \mathrm{MS})$ are taken to facilitate asymptotic normal approximations,

$$\begin{aligned} Q_1 &= \log(\pi) - \log(1-\pi) \\ Q_2 &= \log(c - u_m - \mathrm{MTF}_\phi) \\ Q_3 &= \log(\mathrm{MS})\,. \end{aligned}$$

The form of the estimated asymptotic variance for transformed metric Q_i with GPD estimation is

$$\hat{\mathrm{AV}}_i = \boldsymbol{g}_i^\top(\hat{\sigma}, \hat{\xi})\, \hat{\boldsymbol{\Sigma}}\, \boldsymbol{g}_i(\hat{\sigma}, \hat{\xi})\,.$$

Here $\hat{\boldsymbol{\Sigma}}$ is the estimated asymptotic covariance matrix of the MLEs $(\hat{\sigma}, \hat{\xi})$, obtained from the `fpot` function in the `R` package `evd`, and

$$\begin{aligned} \boldsymbol{g}_1^\top(\sigma,\xi) &= \frac{1}{\pi(1-\pi)}\left(\frac{(c-u_m)\mathrm{MS}}{\sigma} \quad -\frac{\pi(\log(\pi)-\log(1-p_{u_m}))+(c-u_m)\mathrm{MS}}{\xi}\right) \\ \boldsymbol{g}_2^\top(\sigma,\xi) &= \left(\frac{1}{\sigma} \quad \frac{(\xi-\sigma/(\mathrm{MTF}_\phi - c + u_m))(\log(1-p_{u_m})-\log(\phi))-1}{\xi}\right) \\ \boldsymbol{g}_3^\top(\sigma,\xi) &= \left(\frac{1}{\sigma}\frac{c-u_m-\sigma}{\xi(c-u_m)+\sigma} \quad -\frac{1}{\xi}\left(\frac{(\xi+1)(c-u_m)}{\xi(c-u_m)+\sigma} + \frac{\log(\mathrm{MS})+\log(\sigma)-\log(1-p_{u_m})}{\xi+1}\right)\right)\,. \end{aligned}$$

For Pareto estimation, the form of the estimated asymptotic variance for transformed metric Q_i is

$$\hat{\mathrm{AV}}_i = \hat{S}^2 \, [g_i(\hat{\xi})]^2 \, .$$

Here $\hat{S}^2$ is the asymptotic variance of $\hat{\xi}$, which is $\hat{\xi}^2/\hat{k} = \hat{\xi}_h^2/\hat{k}$ in the Pareto-I case and $2\,\hat{\xi}^2/\hat{k} = 2/(\hat{k}\,\hat{b}^2)$ in the Pareto-II case. The sensitivities are given by

$$g_1(\xi) = -\frac{1}{\pi(1-\pi)} \frac{\pi(\log(\pi) - \log(1-p_{u_m}))}{\xi}$$

$$g_2(\xi) = \frac{(\mathrm{MTF}_\phi - c)(\log(1-p_{u_m}) - \log(\phi))}{\mathrm{MTF}_\phi - c + u_m}$$

$$g_3(\xi) = -\frac{1 + \log(\mathrm{MS}) + \log(\xi) + \log(c) - \log(1-p_{u_m})}{\xi} \, .$$

Estimates $\hat{Q}_i$ of the transformed metrics Q_i are obtained by substituting the estimated parameters corresponding to the selected estimation approach. An approximate asymptotic $100(1-\alpha)\%$ confidence interval for Q_i is obtained by

$$\left(\, \hat{Q}_i - z_{\alpha/2} \sqrt{\hat{\mathrm{AV}}_i} \, , \; \hat{Q}_i + z_{\alpha/2} \sqrt{\hat{\mathrm{AV}}_i} \, \right) \, ,$$

where $z_{\alpha/2}$ is the upper $\alpha/2$ quantile of the standard normal distribution. The inverse transforms are applied to the endpoints of these intervals to obtain approximate asymptotic $100(1-\alpha)\%$ confidence intervals for the original metrics $(\pi, \mathrm{MTF}_\phi, \mathrm{MS})$. Approximate asymptotic standard errors for the estimates of the original metrics are also obtained in straightforward fashion,

$$\mathrm{se}(\hat{\pi}) = \hat{\pi}(1-\hat{\pi}) \sqrt{\hat{\mathrm{AV}}_1}$$

$$\mathrm{se}(\hat{\mathrm{MTF}}_\phi) = (c - u_m - \hat{\mathrm{MTF}}_\phi) \sqrt{\hat{\mathrm{AV}}_2}$$

$$\mathrm{se}(\hat{\mathrm{MS}}) = \hat{\mathrm{MS}} \sqrt{\hat{\mathrm{AV}}_3} \, .$$

3.1.1 Example: Using Samples from Pareto Distributed Margins

We continue with the generalized Pareto margins example of Section 2.2. Appendix A.3 provides `R` code that implements the three approaches to estimating the metrics with uncertainty quantification as described in Section 3.1. Appendix A.4 contains all the support code necessary to perform the required calculations. Assume $N = 1000$ samples are taken from the distributions of M_1 and M_2. This results in bootstrap sample sets of size 500 in the double bootstrap procedure, the minimal sample size for which the tail estimation methods were tested.

The estimates of $\hat{k}$ from the shifted M_1 and M_2 samples using the double bootstrap procedure were 264 and 344, giving threshold estimates $\hat{u}_m$ of 2.92331 and 1.44836. The GPD method resulted in scale and tail index estimates $(\hat{\sigma}, \hat{\xi}) = (3.71369, 1.07198)$ and $(2.23524, 1.54971)$ for the POT tail distributions of M_1 and M_2. The corresponding estimated asymptotic covariance matrices $\hat{\mathbf{\Sigma}}$ were found to be

$$\begin{pmatrix} 0.21845 & -0.02968 \\ -0.02968 & 0.01640 \end{pmatrix} \text{ and } \begin{pmatrix} 0.08111 & -0.01972 \\ -0.01972 & 0.02031 \end{pmatrix} .$$

The Pareto-I method produced tail index estimates $\hat{\xi}$ (estimated asymptotic standard errors $\hat{S}$) of 1.16219 (0.07153) and 1.54719 (0.08342) for the POT tails of M_1 and M_2, while for Pareto-II these quantities were 1.12739 (0.09813) and 1.49490 (0.11398).

Table 2 gives estimates of the three metrics (nominal and transformed) and their associated standard errors for margins M_1 and M_2 corresponding to the three methods of estimation discussed in Section 3.1. Recall that the second metric MTF_ϕ is computed assuming $\phi = 0.01$.

Margin	Method	π	MTF_ϕ	MS	Q_1	Q_2	Q_3
M_1	GPD	0.09343	−105	0.00826	−2.27241	4.72111	−4.79598
		(0.00647)	(31)	(0.00049)	(0.07636)	(0.27314)	(0.05933)
	Pareto-I	0.09157	−121	0.00787	−2.29464	4.85446	−4.84431
		(0.00597)	(31)	(0.00003)	(0.07174)	(0.23947)	(0.00362)
	Pareto-II	0.08862	−107	0.00786	−2.33057	4.73778	−4.84659
		(0.00842)	(38)	(0.00006)	(0.10425)	(0.32943)	(0.00797)
M_2	GPD	0.05459	−322	0.00141	−2.85178	5.84507	−6.56497
		(0.00555)	(128)	(0.00009)	(0.10745)	(0.36962)	(0.06173)
	Pareto-I	0.05456	−320	0.00141	−2.85228	5.84028	−6.56405
		(0.00542)	(102)	(0.00006)	(0.10500)	(0.29638)	(0.04536)
	Pareto-II	0.05116	−262	0.00137	−2.92028	5.65440	−6.59408
		(0.00743)	(116)	(0.00009)	(0.15314)	(0.40533)	(0.06906)

Table 2: Estimates and standard errors (in parentheses) of metrics $(\pi, \text{MTF}_\phi, \text{MS})$ and transformed metrics (Q_1, Q_2, Q_3) derived from $N = 1000$ samples of margins M_1 and M_2.

The transformed metrics were used to compute 95% confidence intervals for the true values of the metrics $(\pi, \text{MTF}_\phi, \text{MS})$ using the approach described in Section 3.1. Although unknown in typical applications, these true values for this example are stated in Section 2.2. The resulting intervals for the three estimation methods are provided in Table 3. We observe that these intervals cover the true values with two exceptions for the third metric MS of margin M_1. ♦

		M_1		M_2	
Method	Metric	Lower	Upper	Lower	Upper
GPD	π	0.08151	0.10691	0.04469	0.06653
	MTF_ϕ	−185	−59	−689	−144
	MS	0.00736	0.00928	0.00125	0.00159
Pareto-I	π	0.08052	0.10395	0.04487	0.06621
	MTF_ϕ	−198	−73	−591	−169
	MS	0.00782	0.00793	0.00129	0.00154
Pareto-II	π	0.07345	0.10657	0.03840	0.06785
	MTF_ϕ	−211	−53	−608	−105
	MS	0.00773	0.00798	0.00120	0.00157

Table 3: Lower and upper bounds of 95% confidence intervals for the true values of the metrics $(\pi, \text{MTF}_\phi, \text{MS})$ from each of the three estimation methods. Intervals failing to cover the true value are highlighted in red.

3.1.2 Example: Using Samples from T Distributed Margins

We consider three margins assumed to possess T distributions with different rates of tail decay,

$$\begin{aligned} M_1 &\sim T(\mu_1 = -6, \sigma_1 = 1.5, \xi_1 = 1/3) \\ M_2 &\sim T(\mu_2 = -10, \sigma_2 = 1, \xi_2 = 1) \\ M_3 &\sim T(\mu_3 = -25, \sigma_3 = 0.5, \xi_3 = 1.5)\,. \end{aligned}$$

This notation indicates that $(M_i - \mu_i)/\sigma_i$ is distributed as standard T having $1/\xi_i$ degrees of freedom. This setting is of interest as the T distribution is encountered fairly frequently as a predictive distribution for quantities such as margin in applications where uncertainties in variance parameter(s) are accounted for. The margin M_1 has finite mean and variance, which will typically hold in most applications. The margins M_2 and M_3 have infinite moments analogous to the Pareto cases covered in the previous example. These represent more challenging scenarios for tail index estimation.

Table 4 provides the true values of the three metrics for margins M_1, M_2, and M_3. The second metric MTF_ϕ is computed assuming $\phi = 0.01$.

Margin	π	MTF_ϕ	MS
M_1	0.01400	-0.811	0.00611
M_2	0.03173	-21.8	0.00315
M_3	0.02295	-61.9	0.00061

Table 4: True values of metrics $(\pi, \text{MTF}_\phi, \text{MS})$ associated with distributions of margins M_1, M_2, and M_3.

Sample-based estimates of the assumed POT tail distribution parameter(s) and the three metrics with uncertainty quantification are obtained using the `R` code of Appendix A.5. At first, a sample of size $N = 1000$ was taken from the margin distributions and submitted to the double bootstrap procedure to estimate $\hat{k}$. However, these estimates and the resulting Hill estimates $\hat{\xi}_h$ of the tail index parameter ξ were extremely poor. This was rectified by increasing the sample size to $N = 2500$, and only keeping the largest 40% of the samples from each margin. The result was a sufficiently extensive sample of size $N_{tail} = 1000$ from only the tail being approximated, rectifying the estimation deficiencies arising from the original sample. The value of $p_{u_m} = (N - \hat{k})/N$ is calculated using the full sample size $N = 2500$, while $\hat{k}$ is estimated from the double bootstrap procedure using only the $N_{tail} = 1000$ tail samples.

The estimates of $\hat{k}$ from the shifted M_1, M_2, and M_3 tail samples using the double bootstrap procedure were 148, 260, and 415, giving threshold estimates $\hat{u}_m$ of 2.89523, 2.94011, and 1.15777. The GPD method resulted in scale and tail index estimates $(\hat{\sigma}, \hat{\xi}) = (1.42073, 0.52445)$, $(3.53014, 1.00035)$, and $(1.92308, 1.45803)$ for the POT tail distributions of M_1, M_2, and M_3. The corresponding estimated asymptotic covariance matrices $\hat{\boldsymbol{\Sigma}}$ were found to be

$$\begin{pmatrix} 0.04150 & -0.01457 \\ -0.01457 & 0.01566 \end{pmatrix}, \begin{pmatrix} 0.20864 & -0.03194 \\ -0.03194 & 0.01675 \end{pmatrix}, \text{and} \begin{pmatrix} 0.04532 & -0.01217 \\ -0.01217 & 0.01497 \end{pmatrix}.$$

The Pareto-I method produced tail index estimates $\hat{\xi}$ (estimated asymptotic standard errors $\hat{S}$) of 0.50195 (0.04126), 1.09434 (0.06787), and 1.53661 (0.07543) for the POT tails of M_1, M_2, and

M_3, while for Pareto-II these quantities were 0.52183 (0.06066), 1.02288 (0.08971), and 1.50529 (0.10450). The estimates of ξ for margin M_1 vary substantially from the true value of 1/3. Shifting the margin samples by any amount modifies the estimate of the tail index, removing any expectation that the estimates should resemble the true values used for simulation in this example. This effect is more pronounced for smaller ξ as seen in these results.

Table 5 gives estimates of the three metrics (nominal and transformed) and their associated standard errors for margins M_1, M_2, and M_3 corresponding to the three methods of estimation discussed in Section 3.1. Recall that the second metric MTF_ϕ is computed assuming $\phi = 0.01$.

Margin	Method	π	MTF_ϕ	MS	Q_1	Q_2	Q_3
M_1	GPD	0.01593	−1.492	0.00563	−4.12343	1.42916	−5.17908
		(0.00178)	(0.528)	(0.00049)	(0.11380)	(0.12644)	(0.08635)
	Pareto-I	0.01603	−1.490	0.00572	−4.11737	1.42878	−5.16321
		(0.00172)	(0.519)	(0.00014)	(0.10916)	(0.12427)	(0.02521)
	Pareto-II	0.01684	−1.745	0.00579	−4.06674	1.48796	−5.15227
		(0.00246)	(0.790)	(0.00017)	(0.14861)	(0.17841)	(0.02986)
M_2	GPD	0.03565	−26.4	0.00346	−3.29775	3.50261	−5.66644
		(0.00256)	(5.8)	(0.00022)	(0.07460)	(0.17418)	(0.06354)
	Pareto-I	0.03491	−28.4	0.00328	−3.31957	3.56094	−5.71843
		(0.00236)	(6.1)	(0.00002)	(0.07016)	(0.17221)	(0.00569)
	Pareto-II	0.03234	−22.6	0.00326	−3.39849	3.37828	−5.72717
		(0.00331)	(6.8)	(0.00005)	(0.10586)	(0.23116)	(0.01473)
M_3	GPD	0.02209	−54.3	0.00061	−3.79016	4.35625	−7.40688
		(0.00228)	(18.3)	(0.00004)	(0.10537)	(0.23518)	(0.05856)
	Pareto-I	0.02260	−62.0	0.00059	−3.76712	4.45001	−7.43037
		(0.00221)	(18.4)	(0.00003)	(0.10016)	(0.21478)	(0.04880)
	Pareto-II	0.02168	−54.7	0.00058	−3.80955	4.36079	−7.45127
		(0.00306)	(23.3)	(0.00004)	(0.14445)	(0.29792)	(0.07190)

Table 5: Estimates and standard errors (in parentheses) of metrics $(\pi, \text{MTF}_\phi, \text{MS})$ and transformed metrics (Q_1, Q_2, Q_3) derived from $N = 1000$ tail samples of margins M_1, M_2, and M_3.

The transformed metrics were used to compute 95% confidence intervals for the true values of the metrics $(\pi, \text{MTF}_\phi, \text{MS})$ using the approach described in Section 3.1. Table 4 states these true values which are known for this example. The resulting intervals for the three estimation methods are provided in Table 6. We observe that these intervals cover the true values with three exceptions for the third metric MS of margins M_1 and M_2. ♦

4 When the Margin Depends Upon Performance and Threshold

Let P and T denote the random variables performance and threshold. To analyze system failure, the random variable $M = P - T$ is the margin of interest. When P is independent of T, the pdf of M is

$$f_M(m) = \int_{-\infty}^{\infty} f_T(p-m) f_P(p)\, dp = \int_{-\infty}^{\infty} f_P(t+m) f_T(t)\, dt \tag{15}$$

		M_1		M_2		M_3	
Method	Metric	Lower	Upper	Lower	Upper	Lower	Upper
GPD	π	0.01279	0.01983	0.03095	0.04103	0.01805	0.02702
	MTF_ϕ	-2.666	-0.575	-39.9	-16.8	-100.0	-25.5
	MS	0.00476	0.00667	0.00305	0.00392	0.00054	0.00068
Pareto-I	π	0.01298	0.01977	0.03056	0.03985	0.01864	0.02736
	MTF_ϕ	-2.641	-0.588	-42.6	-18.3	-106.8	-32.6
	MS	0.00545	0.00601	0.00325	0.00332	0.00054	0.00065
Pareto-II	π	0.01264	0.02241	0.02644	0.03951	0.01642	0.02857
	MTF_ϕ	-3.598	-0.438	-39.4	-11.9	-116.8	-20.0
	MS	0.00546	0.00614	0.00316	0.00335	0.00050	0.00067

Table 6: Lower and upper bounds of 95% confidence intervals for the true values of the metrics $(\pi, \text{MTF}_\phi, \text{MS})$ from each of the three estimation methods. Intervals failing to cover the true value are highlighted in red.

and as such is seen to be a convolution of the densities of P and $-T$. The cdf of M is $F_M(m) = \mathbb{P}(P - T \leq m)$ and can be rewritten as

$$\int_{p-t\leq m} f_P(p) f_T(t)\, dp\, dt = \int_{-\infty}^{\infty} f_P(p)\Big(\int_{t=p-m}^{\infty} f_T(t)\, dt\Big) dp = 1 - \int_{-\infty}^{\infty} F_T(p-m) f_P(p)\, dp \quad (16)$$

$$\overset{\text{OR}}{=} \int_{-\infty}^{\infty} f_T(t)\Big(\int_{-\infty}^{p=t+m} f_P(p)\, dp\Big) dt = \int_{-\infty}^{\infty} F_P(t+m) f_T(t)\, dt\,.$$

Recall that the failure probability for a component is $\pi \overset{\text{def}}{=} \mathbb{P}(P > T) \equiv \mathbb{P}(M > 0) = 1 - F_M(0)$, i.e. the probability in the tail to the right of 0; the component reliability is then $1 - \pi$.

Estimators for the three metrics are now given for the case when independent and identically distributed $P_1, \ldots, P_{N_p}$ and $T_1, \ldots, T_{N_t}$ are observed. Let $\boldsymbol{M}$ be the $N_p \times N_t$ matrix of pairwise differences $M_{i,j} = P_i - T_j$:

$$\begin{aligned}
\hat{\pi} &= \frac{1}{N_p N_t} \sum_{i=1}^{N_p} \sum_{j=1}^{N_t} \mathbb{I}\{P_i \geq T_j\} \\
&= \frac{1}{N_p N_t} \sum_{n=1}^{N_p N_t} \mathbb{I}\{vec(\boldsymbol{M})_n > 0\} && \text{sample prop. of positive margins} \\
\hat{Q}_{1-\phi} \text{ such that } & \frac{1}{N_p N_t} \sum_{i=1}^{N_p} \sum_{j=1}^{N_t} \mathbb{I}\Big\{P_i - T_j \leq \hat{Q}_{1-\phi}\Big\} \\
&= \frac{1}{N_p N_t} \sum_{n=1}^{N_p N_t} \mathbb{I}\Big\{vec(\boldsymbol{M})_n \leq \hat{Q}_{1-\phi}\Big\} \approx 1 - \phi && \text{sample } (1-\phi) \text{ quantile} \\
\hat{S} &= \hat{f}_M(0; h_p, h_t) \\
&= \frac{\lambda}{N_t} \sum_{j=1}^{N_t} \hat{f}_P(T_j; h_p) + \frac{1-\lambda}{N_p} \sum_{i=1}^{N_p} \hat{f}_T(P_i; h_t) && \text{kernel density estimate at 0} \qquad (17)
\end{aligned}$$

for $0 \leq \lambda \leq 1$. Further observe that for the third metric estimator

$$\begin{aligned}
\hat{S} &= \frac{\lambda}{N_t} \sum_{j=1}^{N_t} \left(\frac{1}{N_p h_p} \sum_{i=1}^{N_p} K\Big(\frac{T_j - P_i}{h_p}\Big) \right) \; + \; \frac{1-\lambda}{N_p} \sum_{i=1}^{N_p} \left(\frac{1}{N_t h_t} \sum_{j=1}^{N_t} K\Big(\frac{P_i - T_j}{h_t}\Big) \right) \\
&= \frac{\lambda}{N_p N_t h_p} \sum_{i=1}^{N_p} \sum_{j=1}^{N_t} K\Big(\frac{P_i - T_j}{h_p}\Big) \; + \; \frac{1-\lambda}{N_p N_t h_t} \sum_{i=1}^{N_p} \sum_{j=1}^{N_t} K\Big(\frac{P_i - T_j}{h_t}\Big) \\
&= \frac{\lambda}{N_p N_t h_p} \sum_{n=1}^{N_p N_t} K\Big(\frac{vec(\boldsymbol{M})_n}{h_p}\Big) \; + \; \frac{1-\lambda}{N_p N_t h_t} \sum_{n=1}^{N_p N_t} K\Big(\frac{vec(\boldsymbol{M})_n}{h_t}\Big) \qquad (18)
\end{aligned}$$

which shows that it is a weighted combination of two kernel density estimators using $vec(\boldsymbol{M})$ based upon effective sample sizes of N_p and N_t; $\lambda = \frac{N_p}{N_p + N_t}$ is then a natural choice.

All of the above are plug-in estimators based upon (16) or (15). To see this, note that an estimator of the cdf of M is

$$\hat{F}_M(m) = \frac{1}{N_t} \sum_{j=1}^{N_t} \hat{F}_P(T_j + m) = \frac{1}{N_p N_t} \sum_{i=1}^{N_p} \sum_{j=1}^{N_t} \mathbb{I}\left\{P_i \leq T_j + m\right\}$$

The estimator of π is obviously $1 - \hat{F}_M(0)$, and the expression above is the famous nonparametric Wilcoxon-Mann-Whitney statistic. The form of $\hat{Q}_{1-\phi}$ is immediately seen to be derived from (16) as well. The two equivalent forms within (15) give rise to two naive estimators and these are averaged to form $\hat{S}$. The integrals within both forms are replaced by sample means to make (17). The quantity S could also be estimated by using kernel density estimates under the integral within (15); the integral would be performed numerically. Alternatively, it could be formulated using properties of the Fast Fourier Transform (taking care in the implementation to avoid wrap-around effects ([18], Appendix D)). Of course these two alternatives lack the simplicity of the simple plug-in estimator (17, 18) and are unlikely to significantly outperform it in practice. `R` code for computing the three metrics using the methodology of this section is provided in Appendix A.2.

Uncertainties in the estimates of the metrics are again reflected in confidence intervals. Before a more in-depth discussion it is worth pointing out that $vec(\boldsymbol{M})$ is *not* an *iid* sample of size $N_p N_t$; obviously $Cov(M_{i,j}, M_{i',j'}) = Var(P)\mathbb{I}\{i = i'\} + Var(T)\mathbb{I}\{j = j'\} \neq 0$. Therefore one cannot immediately use the confidence intervals of Section 3 with $vec(\boldsymbol{M})$. An analyst *could* however obtain a random sample from M by taking a sample of size $N \leq \min(N_p, N_t)$ from P and T and then forming the differences $M_n = P_n - T_n$ $(n = 1, \ldots, N)$. From here, intervals for the three metrics could be derived using the methodology in Section 3. This is "quick-and-dirty" UQ but does disregard $\big(\max(N_p, N_t) - N\big)$ of the observations. The bootstrap here again seems particularly appealing, but we urge careful implementation and assessment to avoid undesirable properties such as poor frequentist coverage of the confidence intervals. A major contributor to the problem is estimator bias (especially for the second metric $Q_{1-\phi}$), so bias-correction is a necessary step for inferences based upon the bootstrap. This suggests a computationally intensive nested bootstrap, which makes simulation studies far more difficult. Consequently, further investigation is needed at present.

We conclude this section by providing an asymptotic interval for the first and most important metric, the failure probability. There is a whole literature (both parametric and nonparametric) on

the point and interval estimation of π [10]. An interval based upon the unbiased Wilcoxon-Mann-Whitney statistic can be derived using a consistent estimate of $Var(\hat{\pi})$ ([10], pg. 147),

$$\widehat{Var}(\hat{\pi}) = \frac{1}{N_p N_t}\Big[\hat{\pi} + (N_p - 1)\overline{v}_1 + (N_t - 1)\overline{v}_2 - (N_p + N_t - 1)\hat{\pi}^2\Big]$$

$$\overline{v}_1 = \int_{-\infty}^{\infty} [\hat{F}_P(t)]^2 d\hat{F}_T(t) \qquad = \frac{1}{N_p^2 N_t} \sum_{j=1}^{N_t} \Big(\sum_{i=1}^{N_p} \mathbb{I}\{P_i \le T_j\} \Big)^2$$

$$\overline{v}_2 = \int_{-\infty}^{\infty} [1 - \hat{F}_T(p)]^2 d\hat{F}_P(p) = \frac{1}{N_p N_t^2} \sum_{i=1}^{N_p} \Big(\sum_{j=1}^{N_t} \mathbb{I}\{P_i \le T_j\} \Big)^2$$

$$s_N \stackrel{\text{def}}{=} \sqrt{\widehat{Var}(\hat{\pi})}\,,$$

so that a $100(1-\alpha)\%$ confidence interval is given by

$$[\underset{\sim}{\pi}, \widetilde{\pi}] = [\hat{\pi} + \texttt{qnorm}(\alpha/2) s_N,\ \hat{\pi} + \texttt{qnorm}(1-\alpha/2) s_N]\ . \tag{19}$$

5 Conclusions

This report introduced three metrics which can be computed for each component of a multicomponent system. The first metric, component failure probability, is the most obviously pertinent quantity in that it can be plugged directly into inference problems of clear interest such as evaluation of system reliability. The other two metrics are of interest for answering questions such as: How must component margin be shifted if failure rate requirements change? Which components are most susceptible to failure rate change under shifts to their margin distributions?

Nonparametric, asymptotic, and parametric methods were introduced for estimating the three metrics when the margin variable is observed directly. Nonparametric point and interval estimates were shown to be straightforward and easy to compute so long as the metrics were not derived from tail events of the margin distribution (these situations required extreme value theory and had a parametric flavor). We also provided point estimates of the three metrics when the failures depend on the difference between two independent variables ($M = P - T > 0$). As for the uncertainties in this case, we observed that using standard bootstrap resampling led to unsatisfactory results. An asymptotically valid confidence interval was instead provided for the first metric; similar intervals for the other two metrics will require future investigation.

The parametric estimation methods invoke the POT theorem to model the relevant tail of a heavy-tailed margin distribution with a generalized Pareto or Pareto distribution. Samples from the margin distribution are used to estimate the threshold at which the selected analytic distribution describes its tail from a double bootstrap procedure. The parameter(s) of the selected tail distribution are then estimated from the POT samples, and these estimates are plugged into analytic expressions for the three metrics and their standard errors. Transformed versions of the three metrics that facilitate asymptotic normal approximations were used to construct confidence intervals for the unknown true values of the metrics.

It is observed from the examples of Sections 3.1.1 and 3.1.2 that all five failures of the confidence intervals to cover the true values of the three metrics occurred for the third metric, MS. Nevertheless, it is hoped the *ordering* of MS values for different components will be correct for clear distinctions

of magnitude, and this does hold true in these examples. A coverage study was conducted under the simulation settings of Section 3.1.2 to test the performance of the recommended asymptotic confidence intervals for the three metrics computed for margins M_1, M_2, and M_3. Several important observations resulted from this study:

- Uncertainties in the transformed metrics used to construct the asymptotic confidence intervals are generally too optimistic.
- The GPD method produced asymptotic confidence intervals for the three metrics possessing coverage closest to nominal. Pareto-I had the worst performance of the estimation methods.
- All estimation methods produced better-performing asymptotic confidence intervals for the three metrics under heavier-tailed margin distributions.
- Asymptotic confidence intervals for MS using the Pareto-I and Pareto-II estimation methods were rather poor for lighter-tailed margin distributions.

These results are based on 1000 tail observations extracted from the top 40% of margin samples. Actual coverages would be expected to improve with larger sample sets of tail observations, due to the asymptotic nature of these intervals. The superior performance of the GPD estimation method is intuitive from the standpoint that the GPD characterization of the POT theorem is an asymptotic approximation, with the Pareto constituting an additional layer of asymptotic approximation. Again this discrepancy in performance would be expected to diminish as the number of tail observations increases. Based on these results, *we recommend use of the GPD estimation method to compute asymptotic confidence intervals for the three metrics when the estimated tail index takes a value of at least 1.* The nonparametric methods are preferred for lighter-tailed margin distributions, and in any application where margin samples exist to allow estimation of the three metrics (i.e. when the failure probability π or target failure rate ϕ are not too extreme relative to the observed margin samples).

The double bootstrap procedure should be provided with a sufficient amount of margin samples to support a *bootstrap sample size* of at least 500, as the procedure was not tested with smaller bootstrap sample sets. Additionally it is important to note that the double bootstrap procedure may perform better if samples clearly irrelevant to tail estimation are removed, as observed with the T distribution samples in Section 3.1.2. In the event the double bootstrap produces an estimate of $\hat{k}$ too small for the analytical tail approximation to hold for metric estimation, the nonparametric methods must be relied upon to obtain the required estimates. Finally, we note that the parametric estimation methods assume a sample of margin M values is passed to the double bootstrap. In the event the margin must be constructed from independently generated P and T samples, the user would form margin samples $M_i = P_i - T_i$ from independently permuted samples of $\{P\}$ and $\{T\}$.

References

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

[2] Beran, R. and Hall, P. (1993). Interpolated nonparametric prediction intervals and confidence intervals. *Journal of the Royal Statistical Society, Series B*, **55** (3), 643–652.

[3] Buch-Larsen, T., Nielsen, J.P., Guillén, M., and Bolancé C. (2005). Kernel density estimation for heavy-tailed distributions using the Champernowne transformation. *Statistics*, **39** (6), 503–518.

[4] Danielsson, J., de Haan, L., Peng, L., and de Vries, C.G. (2001). Using a bootstrap method to choose the sample fraction in tail index estimation. *Journal of Multivariate Analysis*, **76**, 226–248.

[5] Davison, A.C. and Hinkley, D.V. (1997). *Bootstrap Methods and Their Application*. Cambridge University Press: Cambridge.

[6] Ferguson, T.S. (1996). *A Course in Large Sample Theory*. Chapman & Hall.

[7] Gabaix, X. and Ibragimov, R. (2011). Rank $-$ 1/2: A simple way to improve the OLS estimation of tail exponents. *Journal of Business & Economic Statistics*, **29** (1), 24–39.

[8] Hall, P. (1992). Effect of bias estimation on coverage accuracy of bootstrap confidence intervals for a probability density. *Annals of Statistics*, **20**, 675–694.

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

[10] Kotz, S., Lumelskii, Y., and Pensky, M. (2003). *The Stress-Strength Model and its Generalizations: Theory and Applications*. World Scientific: Singapore.

[11] Makarov, M. (2007). Applications of exact extreme value theorem. *Journal of Operational Risk*, **2** (1), 115–120.

[12] Meeker, W.Q., Hahn, G.J., and Escobar, L.A. (2017). *Statistical Intervals: A Guide for Practitioners and Researchers*, Second Ed., John Wiley & Sons.

[13] Nyblom, J. (1992). Note on interpolated order statistics. *Statistics & Probability Letters*, **14**, 129–131.

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

[15] Qi, Y. (2008). Bootstrap and empirical likelihood methods in extremes. *Extremes*, **11**, 81–97.

[16] Silverman, B.W. (1986). *Density Estimation*. Chapman and Hall: New York.

[17] Wand, M.P., Marron, J.S., and Ruppert, D. (1991). Transformations in density estimation. *Journal of the American Statistical Association*, **86** (414), 343–353.

[18] Wand, M.P. and Jones, M.C. (1994). *Kernel Smoothing*. Chapman and Hall / CRC: Boca Raton.

A R Code for Obtaining the Metrics

A.1 When the Margin Variable is Observed Directly

```
# M_vec is a vector containing iid samples of the margin random variable M
#    a failure occurs when M > 0
# phi is the desired failure probability

Metrics1 <- function(M_vec, phi=0.01){
   M <- M_vec

   ### NOTE: these evaluations really add up within a bootstrap !!!
   #qs <- quantile(M, c(0.25,0.75, 1-phi))
   qs <- quantile(M, c(0.25,0.75, 1-phi), type=9)   # 'type' can matter!

   ### Metric 1:  sample proportion of positive margins (i.e. failures)
   m1 <- mean(M >= 0)

   ### Metric 2:  negative of the (1-phi)th sample quantile
   m2 <- -qs[3]

   ### Metric 3:  kernel density estimate at zero
   #              Gaussian kernels with bandwidth 'bw' chosen by 'bw.nrd'
   #bw <- bw.nrd(M)
   h <- (qs[2] - qs[1])/1.34
   bw <- 1.06 * min(sqrt(var(M)), h) * length(M)^(-1/5)

   m3 <- mean( dnorm(M/bw) ) / bw

   return( c(m1,m2,m3) )
}

CI.proportion <- function(x, conf.level=0.90, Jeffreys=TRUE) {
  N <- length(x)
  nF <- sum(x >= 0)
  alpha <- 1 - conf.level
  if(!Jeffreys)  ans <- c( qbeta(alpha/2, nF,     N-nF+1) ,   qbeta(1-alpha/2, nF+1,   N-nF) )
  if( Jeffreys)  ans <- c( qbeta(alpha/2, nF+0.5, N-nF+0.5) , qbeta(1-alpha/2, nF+0.5, N-nF+0.5) )
  return(ans)
}

# For a description, see
#   http://staff.math.su.se/hoehle/blog/2016/10/23/quantileCI.html
#
# Code taken (and slightly adapted) from
#   https://github.com/hoehleatsu/quantileCI/blob/master/R/quantile_confints.R

CI.quantile <- function(x, p, conf.level=0.95,
                        interpolate=TRUE, fix_interval=TRUE)
{
  x <- sort(x)
  n <- length(x)
  alpha <- 1 - conf.level

  l <- qbinom(  alpha/2, size=n, prob=p)
  l <- l + (isTRUE(all.equal(pbinom(l, prob=p, size=n),alpha/2)))
  u <- qbinom(1-alpha/2, size=n, prob=p) + 1

  ### Adjust (if necessary)
```

```
  l <- max(l,1)
  u <- min(u,n)

  ### Stop here if traditional intervals are desired (no interpolation)
  if (!interpolate) {
    ### If too little coverage => increase length by one (if possible)
    if (fix_interval & (p != 0) & (p != 1)) {
      if ( (pbinom(u, n, p) - pbinom(l - 1, n, p) ) <= 1 - alpha) {
        u <- min(n, u + 1)
      }
    }
    return(x[c(l,u)])
  }

  ### Nyblom (1992) final eq. on pg. 131
  lambda <- function(r, beta, p) {
    pi_r   <- pbinom(r-1, size=n, prob=p) # 1 - pbeta(p, r, N-r+1)
                                          # eq. C.19 in Meeker & Hahn
    pi_rp1 <- pbinom(r,   size=n, prob=p)
    num <-   r  *(1-p)*(pi_rp1 - beta)
    den <- (n-r)*  p  *(beta   - pi_r)
    1 / (1 + (num/den) )
  }
  ci_limit <- function(r,beta) {
    lambda <- lambda(r=r, beta=beta, p=p)
    (1-lambda) * x[r]  +  lambda * x[pmin(r+1,n)]  # can't go beyond n
  }

  ### Nyblom (1992) first eq. on pg. 130
  ans <- c( ci_limit(l,   beta=  alpha/2),
            ci_limit(u-1, beta=1-alpha/2) )
  return(ans)
}

CI.kde0 <- function(x, conf.level=0.90, adjust = FALSE){
  alpha <- 1 - conf.level
  N <- length(x)

  if(!adjust)  h <- bw.nrd(x)

  ### Sain (2003) Eq.6 adjustment
  ### ... this is a TERRIBLE idea using 'h' above as pilot est.
  if( adjust){
    #h <- bw.SJ(x, nb = 1000, method = "dpi")
    h <- bw.SJ(x, nb = 1000, method = "ste")

    m3 <- sum( dnorm(x/h) ) /(N*h)
    h <- dnorm(0) / m3

    #h <- h/2   # go the other direction (undersmooth) ...?
  }

  S1 <- sum( dnorm(x/h) )
  S2 <- sum( dnorm(x/h)^2 )

  m3 <- S1/(N*h)
  tmp1 <- -(m3^2 / N)
  tmp2 <- S2 / (N^2*h^2)
```

```
  sd.hat <- sqrt( tmp1 + tmp2 )
  c( m3 + qnorm(alpha/2)*sd.hat , m3 + qnorm(1-alpha/2)*sd.hat )
}

######################################## Two examples

n = 1000
phi = 0.01

##### M_s ~ Norm(mu, sig)
set.seed(2001)

mu1 <- -2 ;  sig1 <- 1.0
mu2 <- -3 ;  sig2 <- 1.6
y1 <- rnorm(n, mu1, sig1)
y2 <- rnorm(n, mu2, sig2)

#
1 - pnorm(-mu1/sig1)       # the true values
-qnorm(1-phi, mu1, sig1)
dnorm(0, mu1, sig1)

Metrics1(y1, phi)

CI.proportion( y1, conf.level = 0.95, Jeffreys = TRUE )
- CI.quantile( y1, 1-phi, conf.level = 0.95, interpolate = TRUE )   # re-order
CI.kde0(  y1, conf.level = 0.95, adjust = FALSE )

#
1 - pnorm(-mu2/sig2)       # the true values
-qnorm(1-phi, mu2, sig2)
dnorm(0, mu2, sig2)

Metrics1(y2, phi)

CI.proportion( y2, conf.level = 0.95, Jeffreys = TRUE )
- CI.quantile( y2, 1-phi, conf.level = 0.95, interpolate = TRUE )   # re-order
CI.kde0(  y2, conf.level = 0.95, adjust = FALSE )

### M_s ~ GPD(mu, sc, sh)
require(evd)

set.seed(2001)

mu1 = -10 ;  sigma1 = 1 ;    xi1 = 1
mu2 = -25 ;  sigma2 = 0.5 ;  xi2 = 1.5
y1 <- rgpd(n, loc=mu1, scale=sigma1, shape=xi1)
y2 <- rgpd(n, loc=mu2, scale=sigma2, shape=xi2)

#
1-pgpd(0,     loc=mu1, scale=sigma1, shape=xi1)   # the true values
 -qgpd(1-phi, loc=mu1, scale=sigma1, shape=xi1)
  dgpd(0,     loc=mu1, scale=sigma1, shape=xi1)

Metrics1(y1, phi)

CI.proportion( y1, conf.level = 0.95, Jeffreys = TRUE )
- CI.quantile( y1, 1-phi, conf.level = 0.95, interpolate = TRUE )   # re-order
CI.kde0(  y1, conf.level = 0.95, adjust = FALSE )
```

```
#
1-pgpd(0,     loc=mu2, scale=sigma2, shape=xi2)   # the true values
 -qgpd(1-phi, loc=mu2, scale=sigma2, shape=xi2)
  dgpd(0,     loc=mu2, scale=sigma2, shape=xi2)

Metrics1(y2, phi)

CI.proportion( y2, conf.level = 0.95, Jeffreys = TRUE )
- CI.quantile( y2, 1-phi, conf.level = 0.95, interpolate = TRUE )   # re-order
CI.kde0(  y2, conf.level = 0.95, adjust = FALSE )

######################################## Simulation Study

N <- 200
phi <- 0.02

### M ~ Norm(mu, sig)
#mu <- -3
#sig <- 1.6
#
#m1 <- 1 - pnorm(-mu/sig)       # the true values
#m2 <- -qnorm(1-phi, mu, sig)
#m3 <- dnorm(0, mu, sig)

### M ~ GPD(mu, sc, sh)
mu <- -25
sc <- 0.5
sh <- 1.5

m1 <- 1-pgpd(0,     loc=mu, scale=sc, shape=sh)   # the true values
m2 <-  -qgpd(1-phi, loc=mu, scale=sc, shape=sh)
m3 <-   dgpd(0,     loc=mu, scale=sc, shape=sh)

########## Get on with the study
N.it <- 10000
dat <- matrix(NA, N.it, 3)           # est ;  (coverage ; CI length)
DAT <- list(m1=dat, m2=dat, m3=dat)

set.seed(12345)

system.time( #####
for(ii in 1:N.it){

   ### M ~ Norm(mu, sig)
   #M <- rnorm(N, mu,sig)

   ### M ~ GPD(mu, sc, sh)
   M <- rgpd(N, loc=mu, scale=sc, shape=sh)

   est <- Metrics1(M, phi)

   ###
   DAT$m1[ii,1] <- est[1]
   DAT$m2[ii,1] <- est[2]
   DAT$m3[ii,1] <- est[3]

   ###
   CI <- CI.proportion( M, conf.level = 0.95, Jeffreys = TRUE )
```

```
   DAT$m1[ii,2] <- (CI[1] <= m1) & (m1 <= CI[2])
   DAT$m1[ii,3] <- CI[2] - CI[1]

   CI <- CI.quantile( M, 1-phi, conf.level = 0.95,
                      #interpolate = FALSE, fix_interval = TRUE )
                      interpolate = TRUE )
   DAT$m2[ii,2] <- (CI[1] <= -m2) & (-m2 <= CI[2])
   DAT$m2[ii,3] <- CI[2] - CI[1]

   CI <- CI.kde0( M, conf.level = 0.95, adjust = FALSE )
   DAT$m3[ii,2] <- (CI[1] <= m3) & (m3 <= CI[2])
   DAT$m3[ii,3] <- CI[2] - CI[1]
}
) #####

colMeans(DAT$m1, na.rm = TRUE)[1]   # est
colMeans(DAT$m1, na.rm = TRUE)[2]   # coverages
colMeans(DAT$m1, na.rm = TRUE)[3]   # CI lengths

colMeans(DAT$m2, na.rm = TRUE)[1]   # est
colMeans(DAT$m2, na.rm = TRUE)[2]   # coverages
colMeans(DAT$m2, na.rm = TRUE)[3]   # CI lengths

colMeans(DAT$m3, na.rm = TRUE)[1]   # est
colMeans(DAT$m3, na.rm = TRUE)[2]   # coverages
colMeans(DAT$m3, na.rm = TRUE)[3]   # CI lengths
```

A.2 When the Margin Depends Upon Performance and Threshold

```
# P_vec is a vector containing iid samples of the performance random variable P
# T_vec is a vector containing iid samples of the performance random variable T
#    a failure occurs when P > T
# phi is the desired failure probability

Metrics2 <- function(P_vec, T_vec, phi=0.01){
   P <- P_vec
   T <- T_vec
   M <- as.vector( outer(P,T,"-") )

   ### Metric 1:  sample proportion of positive margins (i.e. failures)
   m1 <- mean(M >= 0)

   ### Metric 2:  negative of the (1-phi)th sample quantile
   m2 <- -quantile(M, 1-phi, type=8)

   ### Metric 3:  kernel density estimate at zero
   #              Gaussian kernels with bandwidths chosen by 'bw.nrd'
   hp <- bw.nrd(P)
   ht <- bw.nrd(T)
   est1 <- mean( dnorm(M/hp) ) / hp
   est2 <- mean( dnorm(M/ht) ) / ht

   frac <- length(P) / ( length(P) + length(T) )
   m3 <- (frac)*est1 + (1-frac)*est2

   return( c(m1,m2,m3) )
}
```

A.3 Generalized Pareto Margins

```
# Set seed

set.seed(2001)

# Source support functions

source("../func.R")

# Simulation variables

n = 1000
eps = 0.1
B = 1000
phi = 0.01

# Pareto

mu1 = -10; sigma1 = 1; xi1 = 1;
mu2 = -25; sigma2 = 0.5; xi2 = 1.5;

require(evd)

y1 = sort(rgpd(n,loc=mu1,scale=sigma1,shape=xi1))
c1 = -y1[1]+.01
y1 = y1 + c1
y2 = sort(rgpd(n,loc=mu2,scale=sigma2,shape=xi2))
c2 = -y2[1]+.01
y2 = y2 + c2

detach()

# Double Bootstrap

require(foreach)
require(doParallel)
registerDoParallel(cores=2)

db1 = Q2(y1,eps,B)
var.db1 = db1$xihat^2/db1$khat
db2 = Q2(y2,eps,B)
var.db2 = db2$xihat^2/db2$khat

save.image()

#> db1
#$khat
#[1] 264
#
#$xihat
#[1] 1.162191

#> db2
#$khat
#[1] 344
#
#$xihat
#[1] 1.547189
```

```
# GPD estimation

require(evd)

gpd1 = fpot(y1, y1[n-db1$khat], std.err=TRUE, corr=TRUE)
gpd2 = fpot(y2, y2[n-db2$khat], std.err=TRUE, corr=TRUE)

detach()

save.image()

#> gpd1$estimate
#   scale    shape
#3.713690 1.071976

#> gpd2$estimate
#   scale    shape
#2.235235 1.549708

# Rank - 1/2 regression

econ1 = Q4(y1, db1$khat)
var.econ1 = 2*econ1$xihat^2/econ1$khat
econ2 = Q4(y2, db2$khat)
var.econ2 = 2*econ2$xihat^2/econ2$khat

save.image()

#> econ1
#$khat
#[1] 264
#
#$xihat
#[1] 1.127385

#> econ2
#$khat
#[1] 344
#
#$xihat
#[1] 1.494897

#
# Calculate QMU Metrics
#

pi1 = 0.09090909; mtf1 = -89; ms1 = 0.008264463;
pi2 = 0.05573452; mtf2 = -308; ms2 = 0.001466698;

# Double Bootstrap

phat = (n-db1$khat)/n; um = y1[n-db1$khat];
qmuDB1 = Q6(phat, phi, c1, um, db1$xihat, std.err=TRUE,
            sigma=var.db1)

#> qmuDB1
#$qmu
#[1]  9.156786e-02 -1.212277e+02  7.873084e-03
```

```
#
#$t_qmu
#[1] -2.294640  4.854463 -4.844305
#
#$se_qmu
#[1] 5.967374e-03 3.072697e+01 2.852513e-05
#
#$se_t_qmu
#[1] 0.07173774 0.23947121 0.00362312

phat = (n-db2$khat)/n; um = y2[n-db2$khat];
qmuDB2 = Q6(phat, phi, c2, um, db2$xihat, std.err=TRUE,
            sigma=var.db2)

#> qmuDB2
#$qmu
#[1]  5.456384e-02 -3.203137e+02  1.410159e-03
#
#$t_qmu
#[1] -2.852275  5.840276 -6.564053
#
#$se_qmu
#[1] 5.416805e-03 1.019187e+02 6.396235e-05
#
#$se_t_qmu
#[1] 0.10500407 0.29638377 0.04535826

# confidence intervals
alpha = 0.025
z = qnorm(1-alpha)

tmp = c(qmuDB1$t_qmu[1] - z*qmuDB1$se_t_qmu[1],
        qmuDB1$t_qmu[1] + z*qmuDB1$se_t_qmu[1])
ciDB1 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuDB1$t_qmu[2] + z*qmuDB1$se_t_qmu[2],
        qmuDB1$t_qmu[2] - z*qmuDB1$se_t_qmu[2])
ciDB1 = rbind(ciDB1, c1 - y1[n-db1$khat] - exp(tmp))

tmp = c(qmuDB1$t_qmu[3] - z*qmuDB1$se_t_qmu[3],
        qmuDB1$t_qmu[3] + z*qmuDB1$se_t_qmu[3])
ciDB1 = rbind(ciDB1, exp(tmp))

#> ciDB1
#               [,1]          [,2]
#ciDB1  8.052441e-02   0.103954636
#      -1.980816e+02 -73.162803402
#       7.817374e-03   0.007929191 # does not contain ms1

tmp = c(qmuDB2$t_qmu[1] - z*qmuDB2$se_t_qmu[1],
        qmuDB2$t_qmu[1] + z*qmuDB2$se_t_qmu[1])
ciDB2 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuDB2$t_qmu[2] + z*qmuDB2$se_t_qmu[2],
        qmuDB2$t_qmu[2] - z*qmuDB2$se_t_qmu[2])
ciDB2 = rbind(ciDB2, c2 - y2[n-db2$khat] - exp(tmp))

tmp = c(qmuDB2$t_qmu[3] - z*qmuDB2$se_t_qmu[3],
        qmuDB2$t_qmu[3] + z*qmuDB2$se_t_qmu[3])
```

```
ciDB2 = rbind(ciDB2, exp(tmp))

#> ciDB2
#                [,1]          [,2]
#ciDB2  4.486995e-02  6.620686e-02
#      -5.911659e+02 -1.688006e+02
#       1.290206e-03  1.541264e-03

save.image()

# GPD Estimation

phat = (n-db1$khat)/n; um = y1[n-db1$khat];
qmuGPD1 = Q5(phat, phi, c1, um, gpd1$estimate[1], gpd1$estimate[2],
             std.err=TRUE, sigma=gpd1$var.cov)

#> qmuGPD1
#$qmu
#[1]  9.343410e-02 -1.052083e+02  8.262908e-03
#
#$t_qmu
#[1] -2.272407  4.721106 -4.795979
#
#$se_qmu
#[1] 6.468039e-03 3.067113e+01 4.902426e-04
#
#$se_t_qmu
#[1] 0.07636033 0.27313639 0.05933052

phat = (n-db2$khat)/n; um = y2[n-db2$khat];
qmuGPD2 = Q5(phat, phi, c2, um, gpd2$estimate[1], gpd2$estimate[2],
             std.err=TRUE, sigma=gpd2$var.cov)

#> qmuGPD2
#$qmu
#[1]  5.458929e-02 -3.219661e+02  1.408862e-03
#
#$t_qmu
#[1] -2.851782  5.845070 -6.564973
#
#$se_qmu
#[1] 5.545168e-03 1.277121e+02 8.697170e-05
#
#$se_t_qmu
#[1] 0.10744514 0.36961597 0.06173189

# confidence intervals
tmp = c(qmuGPD1$t_qmu[1] - z*qmuGPD1$se_t_qmu[1],
        qmuGPD1$t_qmu[1] + z*qmuGPD1$se_t_qmu[1])
ciGPD1 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuGPD1$t_qmu[2] + z*qmuGPD1$se_t_qmu[2],
        qmuGPD1$t_qmu[2] - z*qmuGPD1$se_t_qmu[2])
ciGPD1 = rbind(ciGPD1, c1 - y1[n-db1$khat] - exp(tmp))

tmp = c(qmuGPD1$t_qmu[3] - z*qmuGPD1$se_t_qmu[3],
        qmuGPD1$t_qmu[3] + z*qmuGPD1$se_t_qmu[3])
ciGPD1 = rbind(ciGPD1, exp(tmp))
```

```
#> ciGPD1
#                 [,1]          [,2]
#ciGPD1  8.150510e-02   0.106905818
#       -1.847140e+02 -58.659907534
#        7.355813e-03   0.009281863

tmp = c(qmuGPD2$t_qmu[1] - z*qmuGPD2$se_t_qmu[1],
        qmuGPD2$t_qmu[1] + z*qmuGPD2$se_t_qmu[1])
ciGPD2 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuGPD2$t_qmu[2] + z*qmuGPD2$se_t_qmu[2],
        qmuGPD2$t_qmu[2] - z*qmuGPD2$se_t_qmu[2])
ciGPD2 = rbind(ciGPD2, c2 - y2[n-db2$khat] - exp(tmp))

tmp = c(qmuGPD2$t_qmu[3] - z*qmuGPD2$se_t_qmu[3],
        qmuGPD2$t_qmu[3] + z*qmuGPD2$se_t_qmu[3])
ciGPD2 = rbind(ciGPD2, exp(tmp))

#> ciGPD2
#                 [,1]          [,2]
#ciGPD2  4.468640e-02  6.653389e-02
#       -6.894553e+02 -1.438813e+02
#        1.248309e-03  1.590064e-03

save.image()

# Rank - 1/2 regression

phat = (n-econ1$khat)/n; um = y1[n-econ1$khat];
qmuRK1 = Q6(phat, phi, c1, um, econ1$xihat, std.err=TRUE,
            sigma=var.econ1)

#> qmuRK1
#$qmu
#[1]  8.862281e-02 -1.070958e+02  7.855119e-03
#
#$t_qmu
#[1] -2.330568  4.737775 -4.846590
#
#$se_qmu
#[1] 8.419884e-03 3.761389e+01 6.259951e-05
#
#$se_t_qmu
#[1] 0.104246729 0.329426525 0.007969264

phat = (n-econ2$khat)/n; um = y2[n-econ2$khat];
qmuRK2 = Q6(phat, phi, c2, um, econ2$xihat, std.err=TRUE,
            sigma=var.econ2)

#> qmuRK2
#$qmu
#[1]  5.116021e-02 -2.619859e+02  1.368446e-03
#
#$t_qmu
#[1] -2.920278  5.654404 -6.594079
#
#$se_qmu
#[1] 7.433921e-03 1.157406e+02 9.450138e-05
#
```

```
#$se_t_qmu
#[1] 0.15314146 0.40533023 0.06905742

#confidence intervals
tmp = c(qmuRK1$t_qmu[1] - z*qmuRK1$se_t_qmu[1],
        qmuRK1$t_qmu[1] + z*qmuRK1$se_t_qmu[1])
ciRK1 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuRK1$t_qmu[2] + z*qmuRK1$se_t_qmu[2],
        qmuRK1$t_qmu[2] - z*qmuRK1$se_t_qmu[2])
ciRK1 = rbind(ciRK1, c1 - y1[n-econ1$khat] - exp(tmp))

tmp = c(qmuRK1$t_qmu[3] - z*qmuRK1$se_t_qmu[3],
        qmuRK1$t_qmu[3] + z*qmuRK1$se_t_qmu[3])
ciRK1 = rbind(ciRK1, exp(tmp))

#> ciRK1
#               [,1]          [,2]
#ciRK1  7.344834e-02   0.106571729
#      -2.106859e+02 -52.782074959
#       7.733379e-03   0.007978775 # does not contain ms1

tmp = c(qmuRK2$t_qmu[1] - z*qmuRK2$se_t_qmu[1],
        qmuRK2$t_qmu[1] + z*qmuRK2$se_t_qmu[1])
ciRK2 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuRK2$t_qmu[2] + z*qmuRK2$se_t_qmu[2],
        qmuRK2$t_qmu[2] - z*qmuRK2$se_t_qmu[2])
ciRK2 = rbind(ciRK2, c2 - y2[n-econ2$khat] - exp(tmp))

tmp = c(qmuRK2$t_qmu[3] - z*qmuRK2$se_t_qmu[3],
        qmuRK2$t_qmu[3] + z*qmuRK2$se_t_qmu[3])
ciRK2 = rbind(ciRK2, exp(tmp))

#> ciRK2
#               [,1]          [,2]
#ciRK2  3.840411e-02  6.785431e-02
#      -6.084064e+02 -1.054601e+02
#       1.195215e-03  1.566786e-03

save.image()
```

A.4 Support functions `func.R1`

```
#
# Hill estimator
#

xiHat = function(ux)
{
  # ux: sorted vector of upper (k+1) order
  #     statistics
  #  k: number of upper order statistics
  #     used in estimation of tail index
  return( mean(log( ux[-1] )) - log( ux[1] ) )
}

#
# Moment estimator
#

miHat = function(ux, i)
{
  # ux: sorted vector of upper (k+1) order
  #     statistics
  #  k: number of upper order statistics
  #     used in estimation of tail index
  #  i: moment
  return( mean((log( ux[-1] ) - log( ux[1] ))^i) )
}

#
# DB metric
#

mHat = function(ux)
{
  # ux: sorted vector of upper (k+1) order
  #     statistics
  return( miHat(ux, 2) - 2*(xiHat(ux))^2 )
}

#
# Quantile estimator
#

q = function(ux, T, j, k, xik)
{
  # ux: sorted vector of upper (T+1) order
  #     statistics (T >= j, T > k)
  if(is.null(xik)){ xik = xiHat( ux[-(1:(T-k))] ) }
  lq = xik * (log(k) - log(j)) +
       log(ux[T+2-k])
  return( exp(lq) )
}

#
# KS distance
#

ks = function(ux, T, k, xik)
```

```
{
  # ux: sorted vector of upper (T+1) order
  #     statistics (T >= j, T > k)
  vks = NULL
  for( j in 1:T ){
    vks = c(vks, abs( ux[T+1-j] - q(ux, T, j, k, xik) ))
  }
  return( max(vks) )
}

#
# Q1 metric
#

Q1 = function(ux, T, xik=NULL)
{
  # ux: sorted vector of upper (T+1) order
  #     statistics (T >= j, T > k)
  mq = NULL
  for( k in 1:(T-1) ){
    mq = c(mq, ks(ux, T, k, xik))
  }
  return( max(which(mq == min(mq))) )
}

#
# Double Bootstrap
#

#
# Required R packages
#
#require(foreach)
#require(doParallel)
#registerDoParallel(cores=<integer>)
#

# reference class
rc = setRefClass("rc", fields = list(x = "vector"))

bmHat = function(n, m, k, rc)
{
  bs = sort(sample(n,m,replace=TRUE))
  ux = rc$x[bs]
  if( k < (m-1) ){ ux = ux[-(1:(m-k-1))] }
  return( mHat(ux)^2 )
}

Q2 = function(x, eps, B)
{
  #   x: ordered sample
  # eps: must choose in (0, 1/2)
  #   B: number of bootstrap samples
  n = length(x)
  m1 = floor( n^(1-eps) )
  rc1 = rc(x = x)
  em = numeric(m1-1)
  for( k in 1:(m1-1) ){
    tmp = NULL
```

```
    tmp = foreach(ii=1:B, .combine='c') %dopar%
          bmHat(n, m1, k, rc1)
    em[k] = mean(tmp)
  }
  r1 = max(which(em == min(em)))
  m2 = floor( m1^2/n )
  em = numeric(m2-1)
  for( k in 1:(m2-1) ){
    tmp = NULL
    tmp = foreach(ii=1:B, .combine='c') %dopar%
          bmHat(n, m2, k, rc1)
    em[k] = mean(tmp)
  }
  r2 = max(which(em == min(em)))
  rhohat = log(r1)
  rhohat = rhohat/(-2*log(m1) + 2*rhohat)
  khat = floor(r1^2*(1-1/rhohat)^(1/(2*rhohat-1))/r2)
  if (khat < (n-1)) { xihat = xiHat(x[-(1:(n-khat-1))]) }
  else { khat = NA; xihat = NA; }
  return( list(khat=khat, xihat=xihat) )
}

#
# MLE of GEV distribution
#

mlgev = function(y)
{
  # y: vector of observations
  require(evd)
  mlgev = fgev(y,std.err=FALSE)
  detach()
  return( mlgev$estimate )
}

#
# Nemeth and Zempleni (2018) Regression Algorithm
#

xihat_ks = function(uy, T)
{
  khat = Q1(uy, T)
  uy = uy[-(1:(T-khat))]
  return( xiHat(uy) )
}

Q3 = function(x, eps, B, coef_mr, gev=FALSE, coef_rf=NULL)
{
  #       x: ordered sample
  #     eps: must choose in (1/2, 1)
  #       B: number of bootstrap samples
  # coef_mr: regression coefficient vector for bootstrap mean
  #  gev: use GEV location estimate?
  # coef_rf: regression coefficient vector for GEV location
  #          estimate (must be specified if gev=TRUE)
  n = length(x)
  m = floor( n^eps )
  Y = foreach(ii=1:B, .combine='cbind') %dopar%
      x[sort(sample(n,m,replace=TRUE))]
```

```
  T = floor( 0.15 * m )
  Yu = Y[-(1:(m-T-1)),]
  xi = foreach(yu=iter(Yu, by='column'), .combine='c') %dopar%
       xihat_ks(yu, T)
  xihat_mr=coef_mr[1]+coef_mr[2]*mean(xi)
  if( gev ){
    xihat_rf=coef_rf[1]+coef_rf[2]*as.numeric(mlgev( xi )[1])
  }
  T = floor( 0.15 * n )
  x = x[-(1:(n-T-1))]
  q3out = list()
  q3out$khat_mr=Q1(x, T, xik=xihat_mr)
  q3out$xihat_mr=xihat_mr
  if( gev ){
    q3out$khat_rf=Q1(x, T, xik=xihat_rf)
    q3out$xihat_rf=xihat_rf
  }
  return( q3out )
}

#
# Gabaix and Ibragimov Rank - 1/2 regression
#

Q4 = function(x, k)
{
  # x: ordered sample
  # k: number of upper order statistics
  #    used for fitting
  n = length(x)
  rank = seq(k,1,by=-1)
  lm_rr = lm(log(rank - 1/2) ~ log(x[-(1:(n-k))]))
  return( list(khat=k, xihat=-1/as.numeric(lm_rr$coef[2])) )
}

#
# QMU Metrics
#

# GPD

Q5 = function(phat, phi, c, loc, scale, shape,
              std.err=FALSE, sigma=NULL)
{
  #    phat: estimate of margin CDF value at loc
  #     phi: failure probability to compute second metric
  #       c: additive constant to margin samples to
  #          ensure loc > 0
  #     loc: lower bound of GPD tail approximation (loc > 0)
  #   scale: scale parameter of GPD distribution
  #   shape: tail index parameter of GPD distribution
  # std.err: compute standard errors of metrics (if TRUE)
  #   sigma: asymptotic covariance matrix of MLEs (provided
  #          only if std.err=TRUE)
  require(evd)
  qmu = numeric(3)
  if( loc <= c ) {
    qmu[1] = (1-phat) * (1-pgpd(c, loc, scale, shape))
    qmu[3] = (1-phat) * dgpd(c, loc, scale, shape)
```

```
  } else { qmu[1] = NA; qmu[3] = NA; }
  if( phi > 0 && phat+phi <= 1 ){
    qmu[2] = c - qgpd((1-phat-phi)/(1-phat), loc, scale, shape)
  } else { qmu[2] = NA }
  detach()
  if (std.err){
    t_qmu = numeric(3)
    sqmu = t_sqmu = matrix(0,2,3)
    se_qmu = se_t_qmu = numeric(3)
    if( loc <= c ) {
      t_qmu[1] = log(qmu[1]) - log(1-qmu[1])
      sqmu[1,1] = (c-loc)*qmu[3]/scale
      sqmu[2,1] = -(qmu[1]*(log(qmu[1])-log(1-phat))+
                  (c-loc)*qmu[3])/shape
      t_sqmu[,1] = sqmu[,1]/qmu[1]/(1-qmu[1])
      t_qmu[3] = log(qmu[3])
      t_sqmu[1,3] = (c-loc-scale)/(shape*(c-loc)+scale)/scale
      t_sqmu[2,3] = -((shape+1)*(c-loc)/(shape*(c-loc)+scale)+
                     (log(qmu[3])+log(scale)-log(1-phat))/(shape+1))/shape
      sqmu[,3] = t_sqmu[,3]*qmu[3]
      se_qmu[1] = sqrt(t(sqmu[,1]) %*% sigma %*% sqmu[,1])
      se_qmu[3] = sqrt(t(sqmu[,3]) %*% sigma %*% sqmu[,3])
      se_t_qmu[1] = sqrt(t(t_sqmu[,1]) %*% sigma %*% t_sqmu[,1])
      se_t_qmu[3] = sqrt(t(t_sqmu[,3]) %*% sigma %*% t_sqmu[,3])
    } else {
      t_qmu[1] = NA; t_qmu[3] = NA;
      se_qmu[1] = NA; se_qmu[3] = NA;
      se_t_qmu[1] = NA; se_t_qmu[3] = NA;
    }
    if( phi > 0 && phat+phi <= 1 ){
      b = qmu[2]-c+loc
      t_qmu[2] = log(-b)
      t_sqmu[1,2] = 1/scale
      t_sqmu[2,2] = ((shape-scale/b)*(log(1-phat)-log(phi))-1)/shape
      sqmu[,2] = t_sqmu[,2]*b
      se_qmu[2] = sqrt(t(sqmu[,2]) %*% sigma %*% sqmu[,2])
      se_t_qmu[2] = sqrt(t(t_sqmu[,2]) %*% sigma %*% t_sqmu[,2])
    } else {
      t_qmu[2] = NA
      se_qmu[2] = NA; se_t_qmu[2] = NA;
    }
    return( list(qmu=qmu, t_qmu=t_qmu,
                 se_qmu=se_qmu, se_t_qmu=se_t_qmu) )
  } else { return( qmu ) }
}

# Pareto

Q6 = function(phat, phi, c, loc, shape, std.err=FALSE, sigma=NULL)
{
  #    phat: estimate of margin CDF value at loc
  #     phi: failure probability to compute second metric
  #       c: additive constant to margin samples to
  #          ensure loc > 0
  #     loc: lower bound of Pareto tail approximation (loc > 0)
  #   shape: tail index parameter of Pareto distribution
  # std.err: compute standard errors of metrics (if TRUE)
  #   sigma: asymptotic variance of tail index MLE (provided
  #          only if std.err=TRUE)
```

```
  scale=loc*shape
  require(evd)
  qmu = numeric(3)
  if( loc <= c ) {
    qmu[1] = (1-phat) * (1-pgpd(c, loc, scale, shape))
    qmu[3] = (1-phat) * dgpd(c, loc, scale, shape)
  } else { qmu[1] = NA; qmu[3] = NA; }
  if( phi > 0 && phat+phi <= 1 ){
    qmu[2] = c - qgpd((1-phat-phi)/(1-phat), loc, scale, shape)
  } else { qmu[2] = NA }
  detach()
  if (std.err){
    t_qmu = numeric(3)
    sqmu = t_sqmu = numeric(3)
    se_qmu = se_t_qmu = numeric(3)
    if( loc <= c ) {
      t_qmu[1] = log(qmu[1]) - log(1-qmu[1])
      sqmu[1] = -qmu[1]*(log(qmu[1])-log(1-phat))/shape
      t_sqmu[1] = sqmu[1]/qmu[1]/(1-qmu[1])
      t_qmu[3] = log(qmu[3])
      t_sqmu[3] = -(1+log(qmu[3])+log(shape)+log(c)-log(1-phat))/shape
      sqmu[3] = t_sqmu[3]*qmu[3]
      se_qmu[1] = sqrt(sqmu[1] * sigma * sqmu[1])
      se_qmu[3] = sqrt(sqmu[3] * sigma * sqmu[3])
      se_t_qmu[1] = sqrt(t_sqmu[1] * sigma * t_sqmu[1])
      se_t_qmu[3] = sqrt(t_sqmu[3] * sigma * t_sqmu[3])
    } else {
      t_qmu[1] = NA; t_qmu[3] = NA;
      se_qmu[1] = NA; se_qmu[3] = NA;
      se_t_qmu[1] = NA; se_t_qmu[3] = NA;
    }
    if( phi > 0 && phat+phi <= 1 ){
      b = qmu[2]-c+loc
      t_qmu[2] = log(-b)
      sqmu[2] = (qmu[2]-c)*(log(1-phat)-log(phi))
      t_sqmu[2] = sqmu[2]/b
      se_qmu[2] = sqrt(sqmu[2] * sigma * sqmu[2])
      se_t_qmu[2] = sqrt(t_sqmu[2] * sigma * t_sqmu[2])
    } else {
      t_qmu[2] = NA
      se_qmu[2] = NA; se_t_qmu[2] = NA;
    }
    return( list(qmu=qmu, t_qmu=t_qmu,
                 se_qmu=se_qmu, se_t_qmu=se_t_qmu) )
  } else { return( qmu ) }
}
```

A.5 T Margins

```
# Set seed

set.seed(2011)

# Source support functions

source("../func.R")

# Simulation variables

n = 2500
eps = 0.1
B = 1000
phi = 0.01

# T

mu1 = -6; sigma1 = 1.5; xi1 = 1/3;
mu2 = -10; sigma2 = 1; xi2 = 1;
mu3 = -25; sigma3 = 0.5; xi3 = 1.5;

nn = floor(.6 * n)

y1 = sort(sigma1*rt(n,df=1/xi1)+mu1)
y1 = y1[-(1:nn)]
c1 = -y1[1]+.01
y1 = y1 + c1
y2 = sort(sigma2*rt(n,df=1/xi2)+mu2)
y2 = y2[-(1:nn)]
c2 = -y2[1]+.01
y2 = y2 + c2
y3 = sort(sigma3*rt(n,df=1/xi3)+mu3)
y3 = y3[-(1:nn)]
c3 = -y3[1]+.01
y3 = y3 + c3

nt = length(y1)

# Double Bootstrap

require(foreach)
require(doParallel)
registerDoParallel(cores=2)

db1 = Q2(y1,eps,B)
var.db1 = db1$xihat^2/db1$khat
db2 = Q2(y2,eps,B)
var.db2 = db2$xihat^2/db2$khat
db3 = Q2(y3,eps,B)
var.db3 = db3$xihat^2/db3$khat

save.image()

#> db1
#$khat
#[1] 148
#
```

```
#$xihat
#[1] 0.5019466

#> db2
#$khat
#[1] 260
#
#$xihat
#[1] 1.094337

#> db3
#$khat
#[1] 415
#
#$xihat
#[1] 1.536607

# GPD estimation

require(evd)

gpd1 = fpot(y1, y1[nt-db1$khat], std.err=TRUE, corr=TRUE)
gpd2 = fpot(y2, y2[nt-db2$khat], std.err=TRUE, corr=TRUE)
gpd3 = fpot(y3, y3[nt-db3$khat], std.err=TRUE, corr=TRUE)

detach()

save.image()

#> gpd1$estimate
#    scale     shape
#1.4207299 0.5244539

#> gpd2$estimate
#   scale    shape
#3.530140 1.000354

#> gpd3$estimate
#   scale    shape
#1.923083 1.458025

# Rank - 1/2 regression

econ1 = Q4(y1, db1$khat)
var.econ1 = 2*econ1$xihat^2/econ1$khat
econ2 = Q4(y2, db2$khat)
var.econ2 = 2*econ2$xihat^2/econ2$khat
econ3 = Q4(y3, db3$khat)
var.econ3 = 2*econ3$xihat^2/econ3$khat

save.image()

#> econ1
#$khat
#[1] 148
#
#$xihat
#[1] 0.5218309
```

```
#> econ2
#$khat
#[1] 260
#
#$xihat
#[1] 1.022881

#> econ3
#$khat
#[1] 415
#
#$xihat
#[1] 1.505291

#
# Calculate QMU Metrics
#

pi1 = 0.01400423; mtf1 = -0.8110543; ms1 = 0.006108907;
pi2 = 0.03172552; mtf2 = -21.82052; ms2 = 0.003151583;
pi3 = 0.02295079; mtf3 = -61.93002; ms3 = 0.0006119192;

# Double Bootstrap

phat = (n-db1$khat)/n; um = y1[nt-db1$khat];
qmuDB1 = Q6(phat, phi, c1, um, db1$xihat, std.err=TRUE,
            sigma=var.db1)

#> qmuDB1
#$qmu
#[1]  0.016026349 -1.490150432  0.005723283
#
#$t_qmu
#[1] -4.117365  1.428779 -5.163213
#
#$se_qmu
#[1] 0.0017213747 0.5186668335 0.0001442814
#
#$se_t_qmu
#[1] 0.10915845 0.12427319 0.02520955

phat = (n-db2$khat)/n; um = y2[nt-db2$khat];
qmuDB2 = Q6(phat, phi, c2, um, db2$xihat, std.err=TRUE,
            sigma=var.db2)

#> qmuDB2
#$qmu
#[1]   0.03490584 -28.42633759   0.00328488
#
#$t_qmu
#[1] -3.319571  3.560944 -5.718425
#
#$se_qmu
#[1] 2.363357e-03 6.061169e+00 1.868855e-05
#
#$se_t_qmu
#[1] 0.070155468 0.172209858 0.005689265

phat = (n-db3$khat)/n; um = y3[nt-db3$khat];
```

```
qmuDB3 = Q6(phat, phi, c3, um, db3$xihat, std.err=TRUE,
            sigma=var.db3)

#> qmuDB3
#$qmu
#[1]  2.259609e-02 -6.198662e+01  5.929682e-04
#
#$t_qmu
#[1] -3.767123  4.450014 -7.430370
#
#$se_qmu
#[1] 2.211976e-03 1.839086e+01 2.893916e-05
#
#$se_t_qmu
#[1] 0.10015508 0.21477588 0.04880389

# confidence intervals
alpha = 0.025
z = qnorm(1-alpha)

tmp = c(qmuDB1$t_qmu[1] - z*qmuDB1$se_t_qmu[1],
        qmuDB1$t_qmu[1] + z*qmuDB1$se_t_qmu[1])
ciDB1 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuDB1$t_qmu[2] + z*qmuDB1$se_t_qmu[2],
        qmuDB1$t_qmu[2] - z*qmuDB1$se_t_qmu[2])
ciDB1 = rbind(ciDB1, c1 - y1[nt-db1$khat] - exp(tmp))

tmp = c(qmuDB1$t_qmu[3] - z*qmuDB1$se_t_qmu[3],
        qmuDB1$t_qmu[3] + z*qmuDB1$se_t_qmu[3])
ciDB1 = rbind(ciDB1, exp(tmp))

#> ciDB1
#              [,1]         [,2]
#ciDB1  0.012979603  0.019773940
#      -2.641216784 -0.587917184
#       0.005447369  0.006013172 # does not contain ms1

tmp = c(qmuDB2$t_qmu[1] - z*qmuDB2$se_t_qmu[1],
        qmuDB2$t_qmu[1] + z*qmuDB2$se_t_qmu[1])
ciDB2 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuDB2$t_qmu[2] + z*qmuDB2$se_t_qmu[2],
        qmuDB2$t_qmu[2] - z*qmuDB2$se_t_qmu[2])
ciDB2 = rbind(ciDB2, c2 - y2[nt-db2$khat] - exp(tmp))

tmp = c(qmuDB2$t_qmu[3] - z*qmuDB2$se_t_qmu[3],
        qmuDB2$t_qmu[3] + z*qmuDB2$se_t_qmu[3])
ciDB2 = rbind(ciDB2, exp(tmp))

#> ciDB2
#               [,1]          [,2]
#ciDB2   0.030558608   0.039846094
#      -42.556810560 -18.343763968
#        0.003248454   0.003321714 # does not contain ms2

tmp = c(qmuDB3$t_qmu[1] - z*qmuDB3$se_t_qmu[1],
        qmuDB3$t_qmu[1] + z*qmuDB3$se_t_qmu[1])
ciDB3 = exp(tmp)/(1+exp(tmp))
```

```
tmp = c(qmuDB3$t_qmu[2] + z*qmuDB3$se_t_qmu[2],
        qmuDB3$t_qmu[2] - z*qmuDB3$se_t_qmu[2])
ciDB3 = rbind(ciDB3, c3 - y3[nt-db3$khat] - exp(tmp))

tmp = c(qmuDB3$t_qmu[3] - z*qmuDB3$se_t_qmu[3],
        qmuDB3$t_qmu[3] + z*qmuDB3$se_t_qmu[3])
ciDB3 = rbind(ciDB3, exp(tmp))

#> ciDB3
#               [,1]          [,2]
#ciDB3  1.864377e-02  2.736291e-02
#      -1.068055e+02 -3.256659e+01
#       5.388767e-04  6.524892e-04

save.image()

# GPD Estimation

phat = (n-db1$khat)/n; um = y1[nt-db1$khat];
qmuGPD1 = Q5(phat, phi, c1, um, gpd1$estimate[1], gpd1$estimate[2],
             std.err=TRUE, sigma=gpd1$var.cov)

#> qmuGPD1
#$qmu
#[1]  0.015931024 -1.491744320  0.005633166
#
#$t_qmu
#[1] -4.123428  1.429161 -5.179084
#
#$se_qmu
#[1] 0.0017840726 0.5279129266 0.0004864211
#
#$se_t_qmu
#[1] 0.1138003 0.1264403 0.0863495

phat = (n-db2$khat)/n; um = y2[nt-db2$khat];
qmuGPD2 = Q5(phat, phi, c2, um, gpd2$estimate[1], gpd2$estimate[2],
             std.err=TRUE, sigma=gpd2$var.cov)

#> qmuGPD2
#$qmu
#[1]   0.035648608 -26.431942266   0.003460152
#
#$t_qmu
#[1] -3.297746  3.502611 -5.666443
#
#$se_qmu
#[1] 0.0025644787 5.7830678153 0.0002198572
#
#$se_t_qmu
#[1] 0.07459697 0.17417821 0.06353975

phat = (n-db3$khat)/n; um = y3[nt-db3$khat];
qmuGPD3 = Q5(phat, phi, c3, um, gpd3$estimate[1], gpd3$estimate[2],
             std.err=TRUE, sigma=gpd3$var.cov)

#> qmuGPD3
#$qmu
```

```
#[1]  2.209287e-02 -5.432271e+01  6.070639e-04
#
#$t_qmu
#[1] -3.790160  4.356250 -7.406877
#
#$se_qmu
#[1] 2.276439e-03 1.833561e+01 3.554959e-05
#
#$se_t_qmu
#[1] 0.10536738 0.23517982 0.05855988

# confidence intervals
tmp = c(qmuGPD1$t_qmu[1] - z*qmuGPD1$se_t_qmu[1],
        qmuGPD1$t_qmu[1] + z*qmuGPD1$se_t_qmu[1])
ciGPD1 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuGPD1$t_qmu[2] + z*qmuGPD1$se_t_qmu[2],
        qmuGPD1$t_qmu[2] - z*qmuGPD1$se_t_qmu[2])
ciGPD1 = rbind(ciGPD1, c1 - y1[nt-db1$khat] - exp(tmp))

tmp = c(qmuGPD1$t_qmu[3] - z*qmuGPD1$se_t_qmu[3],
        qmuGPD1$t_qmu[3] + z*qmuGPD1$se_t_qmu[3])
ciGPD1 = rbind(ciGPD1, exp(tmp))

#> ciGPD1
#               [,1]         [,2]
#ciGPD1  0.012786808  0.019832856
#       -2.665923084 -0.575295826
#        0.004756108  0.006671959

tmp = c(qmuGPD2$t_qmu[1] - z*qmuGPD2$se_t_qmu[1],
        qmuGPD2$t_qmu[1] + z*qmuGPD2$se_t_qmu[1])
ciGPD2 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuGPD2$t_qmu[2] + z*qmuGPD2$se_t_qmu[2],
        qmuGPD2$t_qmu[2] - z*qmuGPD2$se_t_qmu[2])
ciGPD2 = rbind(ciGPD2, c2 - y2[nt-db2$khat] - exp(tmp))

tmp = c(qmuGPD2$t_qmu[3] - z*qmuGPD2$se_t_qmu[3],
        qmuGPD2$t_qmu[3] + z*qmuGPD2$se_t_qmu[3])
ciGPD2 = rbind(ciGPD2, exp(tmp))

#> ciGPD2
#                [,1]          [,2]
#ciGPD2   0.030949706   0.041030709
#       -39.941577285 -16.829474540
#         0.003054992   0.003919046

tmp = c(qmuGPD3$t_qmu[1] - z*qmuGPD3$se_t_qmu[1],
        qmuGPD3$t_qmu[1] + z*qmuGPD3$se_t_qmu[1])
ciGPD3 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuGPD3$t_qmu[2] + z*qmuGPD3$se_t_qmu[2],
        qmuGPD3$t_qmu[2] - z*qmuGPD3$se_t_qmu[2])
ciGPD3 = rbind(ciGPD3, c3 - y3[nt-db3$khat] - exp(tmp))

tmp = c(qmuGPD3$t_qmu[3] - z*qmuGPD3$se_t_qmu[3],
        qmuGPD3$t_qmu[3] + z*qmuGPD3$se_t_qmu[3])
ciGPD3 = rbind(ciGPD3, exp(tmp))
```

```
#> ciGPD3
#                 [,1]          [,2]
#ciGPD3  1.804502e-02  2.702375e-02
#       -9.997621e+01 -2.552959e+01
#        5.412378e-04  6.808958e-04

save.image()

# Rank - 1/2 regression

phat = (n-econ1$khat)/n; um = y1[nt-econ1$khat];
qmuRK1 = Q6(phat, phi, c1, um, econ1$xihat, std.err=TRUE,
            sigma=var.econ1)

#> qmuRK1
#$qmu
#[1]  0.016844520 -1.744583547  0.005786247
#
#$t_qmu
#[1] -4.066742  1.487956 -5.152271
#
#$se_qmu
#[1] 0.002461173 0.790010479 0.000172798
#
#$se_t_qmu
#[1] 0.14861455 0.17841107 0.02986357

phat = (n-econ2$khat)/n; um = y2[nt-econ2$khat];
qmuRK2 = Q6(phat, phi, c2, um, econ2$xihat, std.err=TRUE,
            sigma=var.econ2)

#> qmuRK2
#$qmu
#[1]   0.032342691 -22.550074077   0.003256293
#
#$t_qmu
#[1] -3.398490  3.378275 -5.727166
#
#$se_qmu
#[1] 3.313205e-03 6.777541e+00 4.798091e-05
#
#$se_t_qmu
#[1] 0.10586457 0.23115643 0.01473482

phat = (n-econ3$khat)/n; um = y3[nt-econ3$khat];
qmuRK3 = Q6(phat, phi, c3, um, econ3$xihat, std.err=TRUE,
            sigma=var.econ3)

#> qmuRK3
#$qmu
#[1]  2.167781e-02 -5.467742e+01  5.807054e-04
#
#$t_qmu
#[1] -3.809550  4.360789 -7.451267
#
#$se_qmu
#[1] 3.063513e-03 2.333271e+01 4.175229e-05
#
```

```
#$se_t_qmu
#[1] 0.14445167 0.29791915 0.07189926

#confidence intervals
tmp = c(qmuRK1$t_qmu[1] - z*qmuRK1$se_t_qmu[1],
        qmuRK1$t_qmu[1] + z*qmuRK1$se_t_qmu[1])
ciRK1 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuRK1$t_qmu[2] + z*qmuRK1$se_t_qmu[2],
        qmuRK1$t_qmu[2] - z*qmuRK1$se_t_qmu[2])
ciRK1 = rbind(ciRK1, c1 - y1[nt-econ1$khat] - exp(tmp))

tmp = c(qmuRK1$t_qmu[3] - z*qmuRK1$se_t_qmu[3],
        qmuRK1$t_qmu[3] + z*qmuRK1$se_t_qmu[3])
ciRK1 = rbind(ciRK1, exp(tmp))

#> ciRK1
#             [,1]         [,2]
#ciRK1  0.01264184  0.022412636
#      -3.59821433 -0.437933001
#       0.00545729  0.006135033

tmp = c(qmuRK2$t_qmu[1] - z*qmuRK2$se_t_qmu[1],
        qmuRK2$t_qmu[1] + z*qmuRK2$se_t_qmu[1])
ciRK2 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuRK2$t_qmu[2] + z*qmuRK2$se_t_qmu[2],
        qmuRK2$t_qmu[2] - z*qmuRK2$se_t_qmu[2])
ciRK2 = rbind(ciRK2, c2 - y2[nt-econ2$khat] - exp(tmp))

tmp = c(qmuRK2$t_qmu[3] - z*qmuRK2$se_t_qmu[3],
        qmuRK2$t_qmu[3] + z*qmuRK2$se_t_qmu[3])
ciRK2 = rbind(ciRK2, exp(tmp))

#> ciRK2
#               [,1]          [,2]
#ciRK2   0.026442593   0.039505847
#      -39.353916595 -11.868189270
#        0.003163597   0.003351705 # does not contain ms2

tmp = c(qmuRK3$t_qmu[1] - z*qmuRK3$se_t_qmu[1],
        qmuRK3$t_qmu[1] + z*qmuRK3$se_t_qmu[1])
ciRK3 = exp(tmp)/(1+exp(tmp))

tmp = c(qmuRK3$t_qmu[2] + z*qmuRK3$se_t_qmu[2],
        qmuRK3$t_qmu[2] - z*qmuRK3$se_t_qmu[2])
ciRK3 = rbind(ciRK3, c3 - y3[nt-econ3$khat] - exp(tmp))

tmp = c(qmuRK3$t_qmu[3] - z*qmuRK3$se_t_qmu[3],
        qmuRK3$t_qmu[3] + z*qmuRK3$se_t_qmu[3])
ciRK3 = rbind(ciRK3, exp(tmp))

#> ciRK3
#               [,1]          [,2]
#ciRK3  1.642046e-02  2.856950e-02
#      -1.167872e+02 -2.003797e+01
#       5.043768e-04  6.685849e-04

save.image()
```