Supplemental document for Multivariate Subgaussian Stable Distributions in R

by Bruce J. Swihart, John P. Nolan

Abstract We introduce and showcase **mvpd**, a package that uses a product distribution (PD) approach to calculating multivariate subgaussian distributions. The family of subgaussian distributions are eliptically contoured multivariate stable distributions that contain the multivariate Cauchy and the multivariate normal distribution.

Thoughts on error propagation in `pmvss`

There are three inexact entities involved in the distribution calculation as found in `mvpd::pmvss()`: the numerical F_G, the numerical f_A and the outer numerical integration.

The outer integral by `integrate` assumes the integrand is calculated without error, but this is not the case. By default, f_A is calculated by **libstableR** to a precision of 1e-12 and the numerical F_G is provided by pmvnorm, whose default precision is 1e-3. So if we requested `abs.tol.si = 1e-4` of the outer integral, we may not be able to trust a successfully returned numerical integration because it is assuming the integrand has 0 error (or relatively negligible error). We can see this in a thought experiment. Let $\widehat{F}_G = F_G + \epsilon_G$ where F_G is without error and ϵ_G we will take as the requested error tolerance of F_G (where ϵ_G=1e-3 as default in `pmvnorm`). Similarly, $\widehat{f}_A = f_A + \epsilon_A$ (where ϵ_A =1e-12 for `libstableR::stable_pdf`). So one could schematically represent our product-distribution integral for F_H as being maximally

$$\begin{aligned}
\widehat{F}_{\mathrm{H}}(\boldsymbol{v},\boldsymbol{w}) &= \int_0^\infty \widehat{f}_B(u)\widehat{F}_G(\boldsymbol{v}/u,\boldsymbol{w}/u)du \\
&\approx \int_0^\infty \left(f_B(u)+2\epsilon_A\right)\left(F_G(\boldsymbol{v}/u,\boldsymbol{w}/u)+\epsilon_G\right)du \\
&\approx \int_0^\infty f_B(u)F_G(\boldsymbol{v}/u,\boldsymbol{w}/u)du + \int_0^\infty \epsilon_G f_B(u)du + \int_0^\infty 2\epsilon_A F_G(\boldsymbol{v}/u,\boldsymbol{w}/u)du + \int_0^\infty 2\epsilon_A\epsilon_G du \\
&= \int_0^\infty f_B(u)F_G(\boldsymbol{v}/u,\boldsymbol{w}/u)du + \epsilon_G + \int_0^\infty 2\epsilon_A F_G(\boldsymbol{v}/u,\boldsymbol{w}/u)du + 2\epsilon_A\epsilon_G \\
&< \int_0^\infty f_B(u)F_G(\boldsymbol{v}/u,\boldsymbol{w}/u)du + \epsilon_G + 2\epsilon_A + 2\epsilon_A\epsilon_G \qquad (1)
\end{aligned}$$

The first term on the right is what we want and the error range is subject to what was specified in `integrate`. Any term involving ϵ_A is inconsequential in the context of what feasible errors can be specified for numerical integration in this context. So looking at ϵ_G (Term 2) is important. For example, suppose the following is returned by `mvpd::pmvss` where `abs.eps.si=1e-4` was specified:

```
0.6904506 with absolute error < 1.1e-05
```

We'd like to be able to interpret that as "we know the distribution is between 0.6903 and 0.6905". However, that's if the integrand was error-less. It is not. To demonstrate this point, we can calculate two scenarios and see where the Scenario 2 of having a sufficiently relatively smaller `abseps.pmvnorm <abs.eps.si` makes the inexactness of our integrand swept under the rug of requested precision of the outer integral:

```
## Scenario 1:
## error propagation
## we truncate the result at 4th decimal place
## and add in the maximal errors
term1 <- trunc(0.6904506*10000)/10000
term1_range <- term1 + c(-1,1)*1e-4
term2 <-  1e-3
term3 <- 2e-12
term4 <- 2e-15
F_H_range <- c(term1_range[1] - term2 - term3 + term4,
               term1_range[2] + term2 + term3 + term4)
```

```
F_H_range <- trunc(F_H_range*10000)/10000

term1_range
F_H_range

round(term1_range,4)
round(F_H_range,4)

## Scenario 2:
term2 <-  1e-6  ## update epsilon_G to 1e-6
term4 <- 2e-18

F_H_range <- c(term1_range[1] - term2 - term3 + term4,
               term1_range[2] + term2 + term3 + term4)
F_H_range <- trunc(F_H_range*1000000)/1000000

term1_range
F_H_range

round(term1_range,4)
round(F_H_range,4)
```

For Scenario 1, the "actual range" due to the imprecision of the integrand could have given a range of $(0.6892, 0.6915)$, not $(0.6903, 0.6905)$. Scenario 2 took care to make sure the integrand was sufficiently more precise than what was being requested of the outer integral and thus the error range from the integral output can be trusted as $(0.6903, 0.6905)$.

It is worthwhile to note how the increasing precision to 1e-6 for abseps.pmvnorm required a 1000-fold increase in `maxpts.pmvnorm`. `mvtnorm::pmvnorm` doesn't error if the maxpts are not sufficient for the requested abs.eps, so `mvpd::pmvss` is written in a way to take note of the failure to get the requested precision and prompt the user to increase `maxpts.pmvnorm` AND/OR increase `abseps.pmvnorm`.