Introduction to the skewtDist package (Version 0.3.0)

Daniel Zhiye Xia

August 26, 2019

Contents

1 Introduction

skewtDist is an R package for two families of skew-t distributions that have different tail behavior for left and right tails, namely the family of the Asymmetric Student t-distributions (AST) distributions introduced by Zhu and Galbraith (2010) (ZG), and the the family of generalized asymmetric t-distributions (GAT) introduced by Baker (2018) (BAK). The importance of these two families is that they go beyond the symmetric tail behaviors of the skew-t distributions derived from skew-normal distributions, as described in Azzalini and Capitanio (2014), and hence can provide better fits for certain data arising in applications, especially for asset returns.

The **skewtDist** package aims to provide a set of tools for modelling non-normally distributed data with skew-t distributions. The package includes, for both AST and GAT families, the following functionality: (a) computation of probability density functions, cumulative distribution functions, quantiles, moments (mean, variance, skewness and excess kurtosis) (b) generation of random variables (c) information matrix computation, and (d) maximum likelihood estimation skew-t distribution parameters. The package may be installed and loaded with the following code.

```
devtools::install_github("dan9401/skewtDist")
library(skewtDist)
```

2 Asymmetric Student t-distribution

2.1 AST Distribution

The AST distribution introduced by Zhu and Galbraith (2010) is generalized from the SST distribution by Hansen (1994) and Fernández & Steel (1998). It is a 5 parameter distribution denoted by $AST(\mu, s, \alpha, \nu_1, \nu_2)$.

- Location parameter μ is the mode, which is generally not the mean of the distribution.
- Scale parameter $s > 0$,which is not the standard deviation.
- The distribution skews to the right when the skewness parameter $\alpha < 0.5$, skews to the left when $\alpha > 0.5$, $\alpha \in (0, 1)$.
- The location parameter μ is equal to the α-th quantile of the distribution.

- The two degrees of freedom / tail parameters each control the left and rights tails of the distribution, separated at the location parameter μ. The left tail parameter $\nu_1 > 0$ controls the left half (below the α-th quantile) of the distribution, while the right tail parameter $\nu_2 > 0$ controls the right half (above the α-th quantile) of the distribution.

The word "asymmetric" in AST indicates not only skewness in the distribution, but also the asymmetry in the two tail powers of the distribution. Its density function has the form[1]:

$$f_{AST}(y;\theta) = \begin{cases} \frac{1}{\sigma}\left[1 + \frac{1}{\nu_1}\left(\frac{y-\mu}{2\alpha\sigma K(\nu_1)}\right)^2\right]^{-(\nu_1+1)/2} & y \leq \mu \\ \frac{1}{\sigma}\left[1 + \frac{1}{\nu_2}\left(\frac{y-\mu}{2(1-\alpha)\sigma K(\nu_2)}\right)^2\right]^{-(\nu_2+1)/2} & y > \mu \end{cases} \tag{1}$$

where $K(\nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\pi\nu}\,\Gamma(\frac{\nu}{2})}$

An AST distribution can be described by specifying the 5 parameters either separately or as a vector, the `dast`, `past`, `qast` and `rast` functions takes in the component parameters or the vectors to calculate the function value or generate random variables.

```
mu <- 0.12; scale <- 0.6; alpha <- 0.6; nu1 <- 5; nu2 <- 6
pars <- c(mu, scale, alpha, nu1, nu2)
# using component parameters
dast(0.12, mu, scale, alpha, nu1, nu2)

## [1] 1.666667

# using parameter vector
dast(0.12, pars = pars)

## [1] 1.666667

past(0.12, pars = pars)

## [1] 0.6

qast(0.12, pars = pars)

## [1] -0.2833839
```

[1]Note that two parameterization of the density function has been introduced in the Zhu and Galbraith (2010) paper, utilizing different scales. Here we present the density function we used to develop the package, see equation (3) in the Appendix for the other parameterization.

Fig.1 shows the density plot of the above AST distribution.

```
# random generation
set.seed(123)
data <- rast(1000, pars = pars)
x <- seq(-3, 3, 0.01)
y <- dast(x, pars = pars)
hist(data, prob = T, breaks = 50)
lines(x, y, col = 4)
```

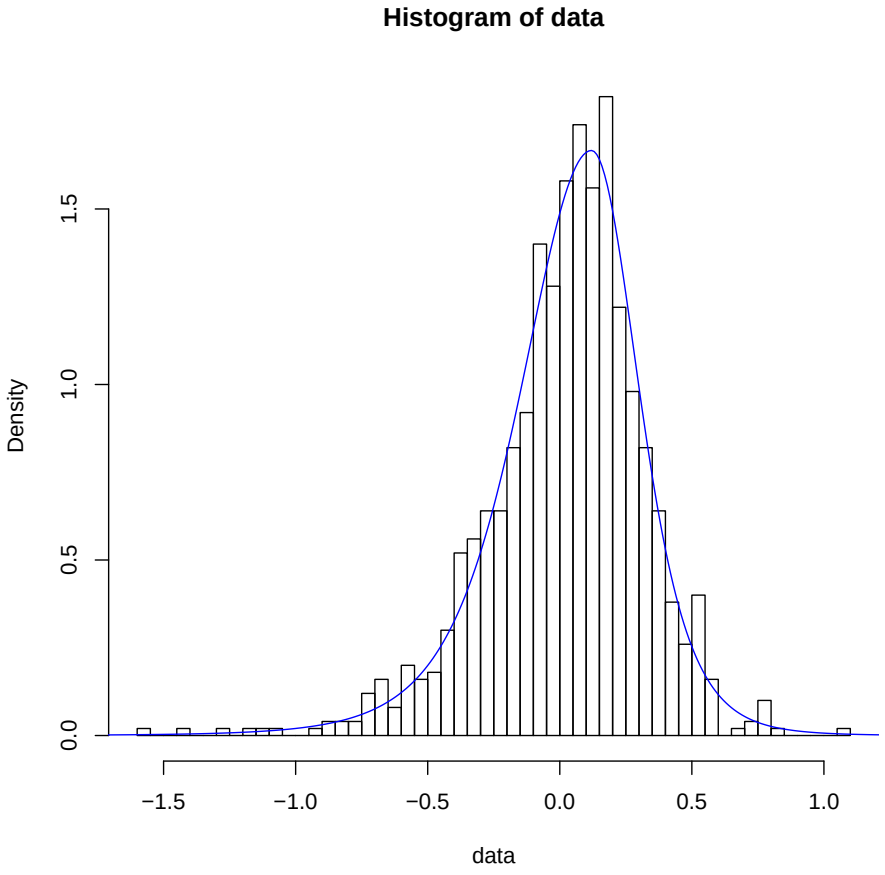

Figure 1: AST Density Plot

2.2 AST Moments

Moments of an AST distribution can be calculated by `astMoments` function. The function returns a vector of mean, standard deviation, skewness and excess kurtosis of an AST distribution. The

`method` argument of the function can be used to choose between analytical calculation or numerical integration. See equations (4)-(7) in the Appendix for details of the analytical formula of AST moments.

```
astMoments(pars = pars, "analytical")

##        mean          sd        skew        kurt
##  0.03187066  0.29527157 -0.98205776  7.10640562

astMoments(pars = pars, "numerical")

##        mean          sd        skew        kurt
##  0.03187066  0.29527157 -0.98205776  7.10640562
```

`astRawMoment` calculates the raw moments and `astCentralMoment` calculates the Central Moments of an AST distribution.

```
astRawMoment(1, pars = pars)

## [1] 0.03187066

astMoments(pars = pars)["mean"] == astRawMoment(1, pars = pars)

## mean
## TRUE

astCentralMoment(2, pars = pars)

## [1] 0.0871853

astMoments(pars = pars)["sd"] == sqrt(astCentralMoment(2, pars = pars))

##   sd
## TRUE
```

2.3 AST Information Matrix

The function `astInfoMat` calculates either the expected information matrix, or the observed information matrix of an AST distribution, and the `method` argument of `astInfoMat` can be used to calculate the expected information matrix (by an asymptotic formula provided in ZG) or observed information matrix (based on derivatives of the log-likelihood evaluated at an observed data set). See Appendix C in the ZG (2009) paper for formulas of expected and observed information matrix.

```
round(astInfoMat(pars = pars, method = "expected"), 3)
```

```
##            mu  sigma   alpha    nu1    nu2
## mu     15.242  0.154 -10.648  0.003 -0.004
## sigma   0.154  3.565  -0.139 -0.029 -0.015
## alpha -10.648 -0.139   9.583 -0.029  0.023
## nu1     0.003 -0.029  -0.029  0.016  0.000
## nu2    -0.004 -0.015   0.023  0.000  0.007
```

`astCov` computes the asymptotic covariance matrix of the parameter MLEs by evaluating the inverse of the information matrix, `astCor` calculates the corresponding asymptotic correlation matrix. The latter provides useful insight on the extent to which pairs of the 5 AST parameter MLEs are correlated with one another.

```
round(astCov(pars = pars), 3)
```

```
##           mu sigma  alpha    nu1     nu2
## mu     0.304 0.001  0.342  0.578  -0.922
## sigma  0.001 0.288  0.006  0.549   0.607
## alpha  0.342 0.006  0.490  0.857  -1.372
## nu1    0.578 0.549  0.857 66.170  -1.241
## nu2   -0.922 0.607 -1.372 -1.241 149.178
```

```
round(astCor(pars = pars), 3)
```

```
##           mu sigma  alpha    nu1    nu2
## mu     1.000 0.003  0.885  0.129 -0.137
## sigma  0.003 1.000  0.015  0.126  0.093
## alpha  0.885 0.015  1.000  0.150 -0.160
## nu1    0.129 0.126  0.150  1.000 -0.012
## nu2   -0.137 0.093 -0.160 -0.012  1.000
```

2.4 astMLE

The function `astMLE` computes the MLE of an AST distribution and returns an `ast` object. The `astMLE` function has the following arguments.

```
args(astMLE)

## function (data, start_pars = c(), fixed_pars = c(), solver = c("nlminb",
##     "nloptr", "Rsolnp"), solver_control = list(), symmetric = FALSE)
## NULL
```

The `start_pars` argument can be specified to define the initial condition of the optimization process, `fixed_pars` can be specified to control the parameters to be kept fixed in the optimization, `symmetry` is a logical argument that tells the function to fit an SST distribution when `TRUE` (see more discussion on this in Section 2.5). In the current version of `astMLE`, the user has the choice of three solvers: `nlminb`, `nloptr` and `Rsolnp`. The solvers `nlminb` and `Rsolnp` are suggested as they are both very stable and yield good solutions. When using `nloptr` as the solver, the algorithms to be used should be specified in the `solver_control` argument. The user should refer to the documentation of the solvers[2] for details of `solver_control`.

```
data(retSW)
ccc <- retSW[,"CCC"]
fit <- astMLE(ccc)
names(fit)

##  [1] "data"            "solver"          "solver_control"
##  [4] "start_pars"      "fixed_pars"      "symmetric"
##  [7] "solver_result"   "fitted_pars"     "objective"
## [10] "time_elapsed"    "message"         "standard_errors"
```

As shown above, the `ast` object contains the input arguments of the `astMLE` function, the fitted parameters, standard errors and other information of the optimization process.

Four S3 method are provided for the `ast` object: `print` which prints the input arguments and solution of the MLE, `summary` adds additional information of the optimization on top of `print`, `moments` which calculates the moments based on the fitted parameters, `plot` which gives the

[2]Packages of the solvers: (a) `nlminb`: `stats` (b) `Rsolnp`: `Rsolnp` (c) `nloptr`: `Nloptr`

option to plot a density plot or QQPlot based on the fitted parameters. Fig.2 shows the QQPlot of the empirical distribution of CCC versus the AST distribution using the fitted parameters.

```
print(fit)

## Distrifitbution:  AST
## Observations:  730
##
## Result:
##                            mu       sigma      alpha       nu1       nu2
## fitted_pars     -0.003710836 0.129378329 0.45622161 3.0554415 4.1264900
## standard_errors  0.004285380 0.004671809 0.02596955 0.1979812 0.2550235

summary(fit)

## Distribution:  AST
## Observations:  730
##
## Result:
##                            mu       sigma      alpha       nu1       nu2
## fitted_pars     -0.003710836 0.129378329 0.45622161 3.0554415 4.1264900
## standard_errors  0.004285380 0.004671809 0.02596955 0.1979812 0.2550235
##
## Log-likelihood -961.6476
##
## Solver:  nlminb
##
##            mu sigma alpha nu1 nu2
## start_pars NA    NA    NA  NA  NA
## fixed_pars NA    NA    NA  NA  NA
##
## Time elapsed:  0.01263404
## Convergence Message:  relative convergence (4)

moments(fit)

##          mean            sd          skew          kurt
##   0.003103676   0.073457927 -10.196380254  26.860549983
```

```
plot(fit, type = "qqplot")
```

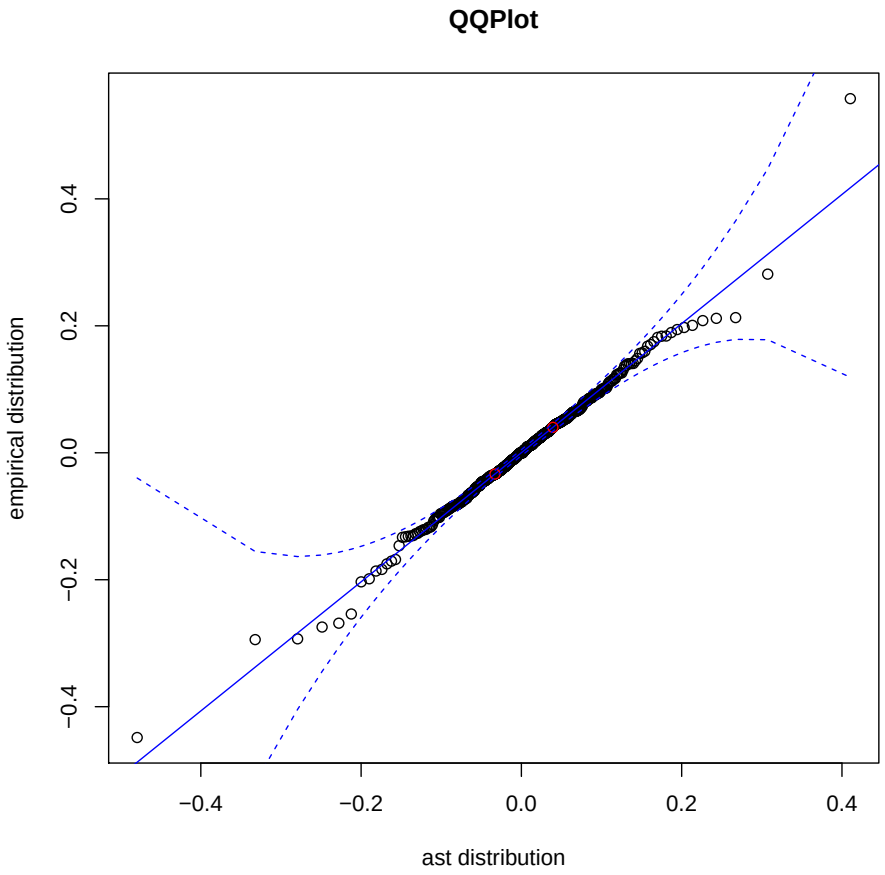

Figure 2: CCC QQPlot-AST

2.5 Symmetric Student t-distribution

As mentioned in Section 2.1, AST distribution is generalized from the SST distribution, such that SST distribution is a special case of the AST distribution. For convenience sake, the package also provides the SST version of the above functions, a typical case would be

```
mu <- 0.12; scale <- 0.6; alpha <- 0.6; nu <- 5
pars <- c(mu, scale, alpha, nu)
dsst(0.12, mu, scale, alpha, nu)

## [1] 1.666667
```

```
dast(0.12, mu, scale, alpha, nu1 = nu, nu2 = nu)

## [1] 1.666667
```

In the case of the MLE functions, `sstMLE` is equivalent to `astMLE` with `symmetric = TRUE`.

```
sstMLE(ccc)$fitted_pars

##           mu        sigma        alpha           nu
## 5.730673e-05 1.279207e-01 4.815548e-01 3.296537e+00

astMLE(ccc, symmetric = T)$fitted_pars

##           mu        sigma        alpha           nu
## 5.730673e-05 1.279207e-01 4.815548e-01 3.296537e+00
```

3 Generalized Asymmetric t-distribution

3.1 GAT Distribution

The GAT distribution introduced by Baker (2018) is a 6 parameter distribution with the notation $GAT(\mu, s, \alpha, r, c, \nu)$.

- Location parameter μ is the center of location, generally not the mean
- Scale parameter $s > 0$, is generally not the standard deviation
- $\alpha > 0$ controls how early tail behavior is apparent
- $r > 0$ controls tail power asymmetry
- $c > 0$ controls the scale asymmetry
- $\nu > 0$ is the tail power / degree of freedom parameter

Density function of GAT distributions has the form[3]

[3] $B(\cdot)$ is the beta function.

$$f(x) = \frac{\alpha(1+r^2)}{r\phi} \frac{\left\{\left[cg((x-\mu)/\phi)\right]^{\alpha r} + \left[cg((x-\mu)/\phi)\right]^{\alpha/r}\right)^{-\nu/\alpha}}{B(\frac{\nu}{\alpha(1+r^2)}, \frac{r^2\nu}{\alpha(1+r^2)})} (1 + ((x-\mu)/\phi)^2)^{-1/2} \tag{2}$$

An GAT distribution can be described by specifying the 6 parameters either separately or as a vector, the `dgat`, `pgat`, `qgat` and `rgat` functions takes in the component parameters or the vectors to calculate the function value or generate random variables.

```
mu <- 0.12; scale <- 0.6; alpha <- 1.5; r <- 1.2; c <- 2; nu <- 5
pars <- c(mu, scale, alpha, r, c, nu)
# using component parameters
dgat(0.12, mu, scale, alpha, r, c, nu)

## [1] 0.1719337

# using parameter vector
dgat(0.12, pars = pars)

## [1] 0.1719337

pgat(0.12, pars = pars)

## [1] 0.9804489

qgat(0.12, pars = pars)

## [1] -0.9530809
```

Fig.3 shows the density plot of the above GAT distribution.

```
# random generation
set.seed(123)
data <- rgat(1000, pars = pars)
x <- seq(-3, 3, 0.01)
y <- dgat(x, pars = pars)
hist(data, prob = T, breaks = 50)
lines(x, y, col = 4)
```

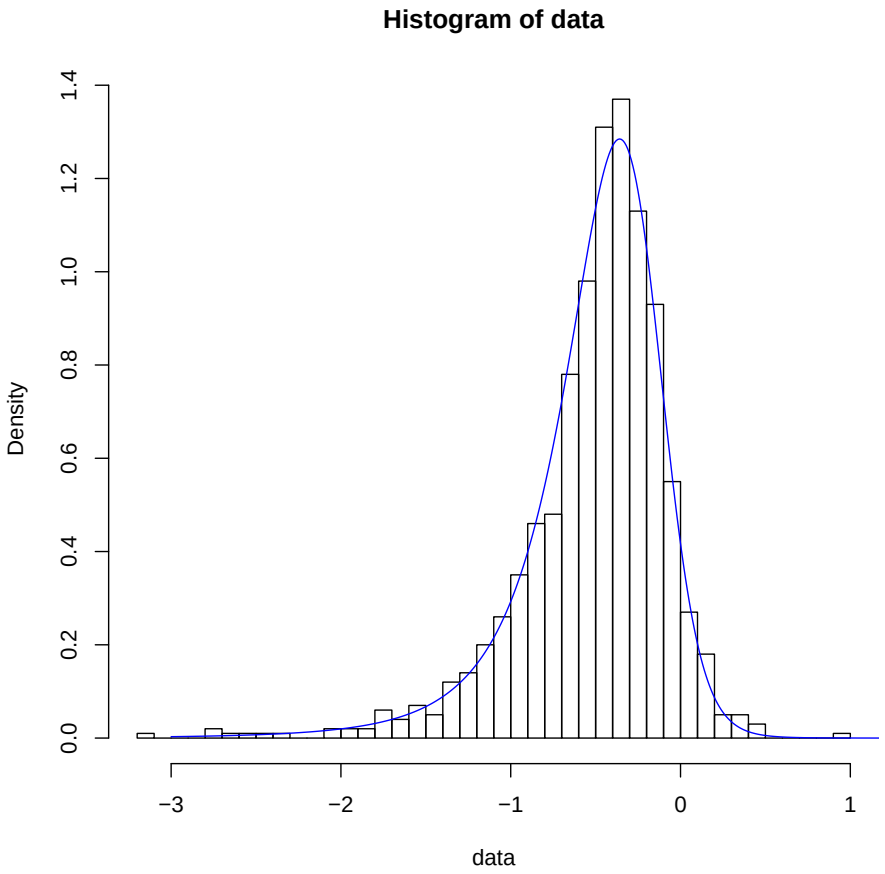

Figure 3: GAT Density Plot

3.2 GAT Moments

Moments of an GAT distribution can be calculated by `gatMoments` function. The function returns a vector of mean, standard deviation, skewness and excess kurtosis of an GAT distribution. The `method` argument of the function can be used to choose between analytical calculation or numerical integration. See equations (8)-(9) in the Appendix for details of the analytical formula of GAT moments.

```
gatMoments(pars = pars, "analytical")

##        mean          sd        skew        kurt
##   0.6394189   0.4462884   3.0609424 130.7064951

gatMoments(pars = pars, "numerical")
```

```
##        mean          sd        skew        kurt
##   0.6394189   0.4462884   3.0609424 130.7064951
```

`gatRawMoment` calculates the raw moments and `gatCentralMoment` calculates the Central Moments of an GAT distribution.

```
gatRawMoment(1, pars = pars)

## [1] 0.6394189

gatMoments(pars = pars)["mean"] == gatRawMoment(1, pars = pars)

## mean
## TRUE

gatCentralMoment(2, pars = pars)

## [1] 0.1991733

gatMoments(pars = pars)["sd"] == sqrt(gatCentralMoment(2, pars = pars))

##   sd
## TRUE
```

3.3 GAT Information Matrix

The function `gatInfoMat` calculates the observed information matrix of an GAT distribution, note that in the current version of the package, the `expected` method of `gatInfoMat` is not yet provided. This is due to the fact that the theoretical framework of the GAT distribution is less complete than that of the AST distribution, it requires derivation of the formulas for the information matrix, and this was not provided by BAK. Considering the complexity in the log-likelihood function and the density function of GAT distribution, this is not provided in the current version, but will be provided in a future update.

```
data <- rgat(1000, pars = pars)
round(gatInfoMat(pars = pars, data = data, method = "observed"), 3)

##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## [1,] 10.520 -6.750  0.128 -5.222 -3.813 -0.158
## [2,] -6.750  7.698 -0.687  4.125  3.047 -0.160
## [3,]  0.128 -0.687  0.114 -0.227 -0.160  0.046
## [4,] -5.222  4.125 -0.227  3.010  2.083  0.002
## [5,] -3.813  3.047 -0.160  2.083  1.501  0.008
## [6,] -0.158 -0.160  0.046  0.002  0.008  0.025
```

`astCov` computes the asymptotic covariance matrix of the parameter MLEs by evaluating the inverse of the information matrix, `astCor` calculates the corresponding asymptotic correlation matrix. The latter provides useful insight on the extent to which pairs of the 5 AST parameter MLEs are correlated with one another.

```
round(gatCov(pars = pars, data = data), 3)

##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,]   21613.34  -26864.95   35073.52  -44958.79   176369.2  -155643.5
## [2,]  -26864.95   35536.29  -40862.67   58631.79  -227155.5   204678.1
## [3,]   35073.52  -40862.67   60575.09  -69471.17   276139.6  -238614.5
## [4,]  -44958.79   58631.79  -69471.17   97056.00  -377060.8   338181.4
## [5,]  176369.22 -227155.49  276139.60 -377060.83  1468587.5 -1311677.9
## [6,] -155643.50  204678.14 -238614.52  338181.42 -1311677.9  1180224.2

round(gatCor(pars = pars, data = data), 3)

##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## [1,]  1.000 -0.969  0.969 -0.982  0.990 -0.975
## [2,] -0.969  1.000 -0.881  0.998 -0.994  0.999
## [3,]  0.969 -0.881  1.000 -0.906  0.926 -0.892
## [4,] -0.982  0.998 -0.906  1.000 -0.999  0.999
## [5,]  0.990 -0.994  0.926 -0.999  1.000 -0.996
## [6,] -0.975  0.999 -0.892  0.999 -0.996  1.000
```

Note that in the above functions, the covariance matrix of the GAT distribution have abnormally large elements, this is likely that there is error in the implementation of the score functions. This will be looked into and fixed in the next update.

3.4 gatMLE

The function `gatMLE` computes the MLE of an GAT distribution and returns an `gat` object. The `gatMLE` takes following parameters

```
args(gatMLE)

## function (data, start_pars = c(), fixed_pars = c(), solver = c("nlminb",
##     "nloptr", "Rsolnp"), solver_control = list())
## NULL
```

The `start_pars` argument can be specified to define the initial condition of the optimization process, `fixed_pars` can be specified to control the parameters to be kept fixed in the optimization. In the current version of `gatMLE`, the user has the choice of three solvers: `nlminb`, `nloptr` and `Rsolnp`. The solvers `nlminb` and `Rsolnp` are suggested as they are both very stable and yield good solutions. When using `nloptr` as the solver, the algorithms to be used should be specified in the `solver_control` argument. The user should refer to the documentation of the solvers[4] for details of `solver_control`.

```
data(retSW)
ccc <- retSW[,"CCC"]
fit <- gatMLE(ccc)
names(fit)

##  [1] "data"            "solver"          "solver_control"
##  [4] "start_pars"      "fixed_pars"      "solver_result"
##  [7] "fitted_pars"     "objective"       "time_elapsed"
## [10] ""                "standard_errors"
```

As shown above, the `gat` object contains the input arguments of the `gatMLE` function, the fitted parameters, standard errors and other important information of the optimization process.

[4]Packages of the solvers: (a) `nlminb`: `stats` (b) `Rsolnp`: `Rsolnp` (c) `nloptr`: `Nloptr`

Four S3 method are provided for the `gat` object: `print` which prints the input arguments and solution of the MLE, `summary` adds additional information of the optimization on top of `print`, `moments` which calculates the moments based on the fitted parameters, `plot` which gives the option to plot a density plot or QQPlot based on the fitted parameters. Fig.4 shows the QQPlot of the empirical distribution of CCC versus the GAT distribution using the fitted parameters.

```
print(fit)

## Distrifitbution:  GAT
## Observations:  730
##
## Result:
##                          mu        phi    alpha         r         c
## fitted_pars     -0.02792033 0.08583634 1.087274 1.2939014 0.5318923
## standard_errors  0.09099041 0.04069586 2.465875 0.5178535 0.4911287
##                        nu
## fitted_pars     3.400093
## standard_errors 1.510513

summary(fit)

## Distribution:  GAT
## Observations:  730
##
## Result:
##                          mu        phi    alpha         r         c
## fitted_pars     -0.02792033 0.08583634 1.087274 1.2939014 0.5318923
## standard_errors  0.09099041 0.04069586 2.465875 0.5178535 0.4911287
##                        nu
## fitted_pars     3.400093
## standard_errors 1.510513
##
## Log-likelihood -961.6153
##
## Solver:  nlminb
##
```

```
##            mu phi alpha  r  c nu
## start_pars  1   1     0  2  1  1
## fixed_pars NA  NA    NA NA NA NA
##
## Time elapsed:  0.04166985
## Convergence Message:
```

```
moments(fit)
```

```
##        mean          sd        skew        kurt
## -0.03098464  0.07521850         NaN         NaN
```

```
plot(fit, type = "qqplot", main = "CCC")
```

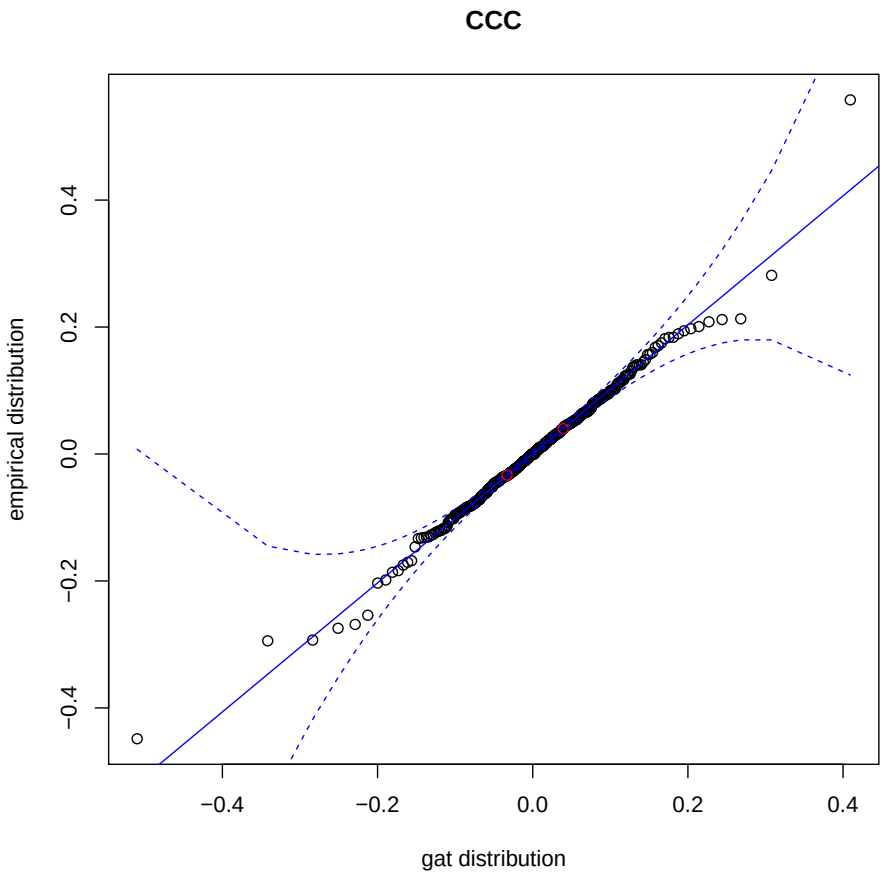

Figure 4: CCC QQPlot-GAT

4 Future Developments

4.1 Initial condition

In the current package implementation, a preset of parameters, regardless of the data set, are provided as the initial conditions of the optimization in the MLE process when `start_pars` is not specified. Having a method of selecting the initial parameters based on the data set can help to achieve higher log-likelihood. Azzalini and Salehi (2019) introduced a set of quantile methods to select the initial parameters, a careful investigation of their strategy and how it can be adopted to AST and GAT distribution will be another focus of next update.

4.2 Hypothesis tests

Both the AST and GAT distribution aims to model skewed fat-tailed data with tail power asymmetry, it is of our interest to design a hypothesis test to determine whether a data set follows a skew-t distribution.

A related test would be determining the goodness of fit of the MLE solutions. As of now, determining goodness of fit relies on visual detection of non-linearity of QQplot. Although the visual checks are useful, there is a need to have a significance test to judge lack of good fit.

4.3 Information Matrices

As of now, the expected and observed information matrices of AST distribution still have different value, which should not be the case theoretically. This is likely caused by an error in the formula or the code implementation, but due to the complexity of the AST distribution, the error has not been found. In future developments, a careful study of the AST as well as GAT (for the reason mentioned in Section 3.3) information matrices will be needed to make sure the expected and information matrices have the same and accurate result.

References

Azzalini, A. (2013). The skew-normal and related families (Vol. 3). Cambridge University Press.

Azzalini, A., & Salehi, M. (2019). Some computational aspects of maximum likelihood estimation of the skew-t distribution. arXiv preprint arXiv:1907.10397.

Baker, R. D. (2016). A new asymmetric generalisation of the t-distribution. arXiv preprint arXiv:1606.05203.

Fernández, C., & Steel, M. F. (1998). On Bayesian modeling of fat tails and skewness. Journal of the American Statistical Association, 93(441), 359-371.

Hansen, B. E. (1994). Autoregressive conditional density estimation. International Economic Review, 705-730.

Zhu, D., & Galbraith, J. (2009). A Generalized Asymmetric Student-t Distribution with Application to Financial Econometrics. CIRANO.

Zhu, D., & Galbraith, J. W. (2010). A generalized asymmetric Student-t distribution with application to financial econometrics. Journal of Econometrics, 157(2), 297-305.

Zhu, D., & Galbraith, J. W. (2011). Modeling and forecasting expected shortfall with the generalized asymmetric Student-t and asymmetric exponential power distributions. Journal of Empirical Finance, 18(4), 765-778.

Appendix

The second parameterization of AST distribution:

$$f_{AST}(y;\theta) = \begin{cases} \frac{\alpha}{\alpha^*} K(\nu_1)\left[1 + \frac{1}{\nu_1}\left(\frac{y-\mu}{2\alpha^*\sigma}\right)^2\right]^{-(\nu_1+1)/2} & y \le \mu \\ \frac{1-\alpha}{1-\alpha^*} K(\nu_2)\left[1 + \frac{1}{\nu_2}\left(\frac{y-\mu}{2(1-\alpha^*)\sigma}\right)^2\right]^{-(\nu_2+1)/2} & y > \mu \end{cases} \tag{3}$$

where $\alpha* = \frac{\alpha K(\nu_1)}{\alpha K(\nu_1)+(1-\alpha)K(\nu_2)}$

In the ZG paper, only the formula of raw moments of standard AST distribution is provided:

$$E(Z^n) = \alpha[-2\alpha^*]^n E|T(\nu_1)|^n + (1-\alpha)[-2(1-\alpha^*)]^n E|T(\nu_2)|^n, \quad n < \nu_1 \wedge \nu_2 \tag{4}$$

where $E|T(\nu_1)|^n$is the n-th absolute moment of standard student t-distribution:

$$E|T(\nu)|^n = \sqrt{\frac{\nu^n}{\pi}}\,\Gamma(\frac{n+1}{2})\,\Gamma(\frac{\nu-n}{2})\,/\,\Gamma(\frac{\nu}{2}), \quad -1 < n < \nu \tag{5}$$

Then we derive the n-th raw moment of general AST:

$$E(X^n) = E((\mu + sZ)^n) = \sum_{i=0}^{n} \binom{n}{i} \mu^i s^{n-i} E(Z^{n-i}) \tag{6}$$

and thus the n-th central moment of general AST:

$$E((X - E(X))^n) = \sum_{i=0}^{n} (-1)^i \binom{n}{i} (E(X))^i E(X^{n-i}) \tag{7}$$

In the BAK paper, centralized moment with respect to the location parameter is provided:

$$E(X-\mu)^n = (\frac{s}{2})^n \frac{\sum_{m=0}^{n} (-1)^m \binom{n}{m} c^{n-m} B\left(\frac{\nu}{\alpha(1+r^2)} - (n-2m)\delta, \frac{\nu r^2}{\alpha(1+r^2)} + (n-2m)\delta\right)}{B(\frac{\nu}{\alpha(1+r^2)}, \frac{\nu r^2}{\alpha(1+r^2)})} \tag{8}$$

where $\delta = \frac{r}{\alpha(1+r^2)}$, and thus we have the n-th raw moment of standard GAT:

$$E(Z^n) = \frac{E(X-\mu)^n}{s^n} = (\frac{1}{2})^n \frac{\sum_{m=0}^{n} (-1)^m \binom{n}{m} c^{n-m} B\left(\frac{\nu}{\alpha(1+r^2)} - (n-2m)\delta, \frac{\nu r^2}{\alpha(1+r^2)} + (n-2m)\delta\right)}{B(\frac{\nu}{\alpha(1+r^2)}, \frac{\nu r^2}{\alpha(1+r^2)})} \tag{9}$$

Then we can deliver the raw and central moment of general GAT by the form of equations (6) and (7).