3 | Modeling Univariate Distributions

The kernel density estimation in the previous chapter is nonparametric estimation. Several parametric distributions have been proposed in the literature for the marginal distributions of financial data. Typically, the parameters in these distributions are estimated by maximum likelihood.

Parametric Models

In a parametric model, the distribution of the data is fully specified except for a finite numbers of unknown parameters. Parameters are often classified as location, scale or shape parameters. A location parameter shifts a distribution to the right or left without changing the distribution's shape or dispersion. Scale parameters quantify dispersion. A parameter λ is a scale parameter of a random variable X if the scale parameter of bX is $|b|\lambda$ for any constant b.

Location-scale families

Let X to be a random variable with a CDF $F(x)$. If a constant a is added to X, the resulting variable $Y = X + a$ has a CDF $F(y - a)$. Suppose that $Y = bX$, where b is a constant, then Y has a CDF $F(y/b)$. Combining these two types of transformations into $Y = a + bX$, the CDF of Y becomes $F\{(y - a)/b\}$ and if F has a density

f, then the density of Y is given by $1/b \cdot f\{(y-a)/b\}$.

An easy way to generate a location-scale family is to start with a standard variable Z such that its CDF F and PDF f, if exists, do not depend on unknown parameters. Often but not always, Z has a location 0 and scale 1. The next step is to consider a random variable $Y = \delta + \theta Z$. Then the CDF and PDF of Y will be

$$F_Y(y) = F\Big(\frac{y-\delta}{\theta}\Big) \quad \text{and} \quad f_Y(y) = \frac{1}{\theta} f\Big(\frac{y-\delta}{\theta}\Big), \qquad \theta > 0\,. \tag{3.1}$$

The totality of such distributions for a fixed F with δ, and θ varying forms a location-scale family.

Eg 3.1. *Gaussian*. The standard PDF $f(z) = \dfrac{1}{\sqrt{2\pi}} \exp\Big\{-\dfrac{z^2}{2}\Big\}$.

The Gaussian location and scale family with mean μ and standard deviation σ has a PDF

$$f_Y(y|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big\{-\frac{(y-\mu)^2}{2\sigma^2}\Big\}\,.$$

Figure **??** shows members of location family, scale family and location-scale family of normal distributions.

Eg 3.2. *Cauchy*. The standard PDF $f(z) = \dfrac{1}{\pi}\dfrac{1}{1+z^2}$.

The Cauchy location and scale family has a PDF

$$f(y|\delta,\theta) = \frac{1}{\pi\theta}\frac{1}{1+\big(\frac{y-\delta}{\theta}\big)^2} = \frac{1}{\pi\theta}\frac{\theta^2}{\theta^2+(y-\delta)^2}.$$

Note that for this family, δ is not the mean and θ is not the standard deviation.

Eg 3.3. *Two parameter exponential*. If $X \sim \text{Exp}(\theta)$, then $Y = X + \alpha$ is called two parameter exponential random variable, denoted by $\text{Exp}(\alpha,\theta), \theta > 0$. Its CDF is

$$f_Y(y|\alpha,\theta) = \begin{cases} \dfrac{1}{\theta}\exp\Big\{-\dfrac{y-\alpha}{\theta}\Big\} & , \quad y \geq \alpha \\ 0 & , \quad y < \alpha \end{cases}.$$

If $Y \sim \text{Exp}(\alpha,\theta)$, then $E(Y) = \alpha + \theta$ and $\text{var}(Y) = \theta^2$.

Moments, Skewness and Kurtosis

The mean and variance of a random variable are cases of moments.

Moments Let X be a random variable, the kth moment of X is $E(X^k)$ and the kth central moment is defined as

$$E[(X-\mu)^k], \quad \text{where} \quad \mu = E[X]\,.$$

The expectation (mean) is the first moment, and variance is the second central moment.

Skewness Skewness measures the degree of asymmetry, with symmetry implying zero skewness. Let Y be a random variable with mean μ and standard deviation σ. If Y has finite third moment, the skewness of Y is its third standardized moment,

$$\text{Sk} = \frac{E(Y-\mu)^3}{\sigma^3}\,.$$

Eg 3.4. *Binomial*. A nice example to show the skewness measure is that of a binomial distribution. If $Y \sim Bin(n,p)$, its skewness is

$$\text{Sk}^{Bin} = \frac{1-2p}{\sqrt{np(1-p)}}.$$

Clearly, $p = 1/2$ gives a symmetric distribution, $p > 1/2$ a left skewed and $p < 1/2$ a right skewed distribution. For any fixed p, the distribution approach symmetry when n increase to ∞.

Kurtosis Kurtosis indicates the extent to which probability is concentrated in the center and especially the tails of the distribution rather than the regions between the center and the tails. If a random variable Y has finite fourth moment, the kurtosis of Y is its fourth standardized moment

$$\text{Kur} = \frac{E(Y-\mu)^4}{\sigma^4},$$

A normal random variable has kurtosis 3. The excess kurtosis is defined as Kur-3 as a comparison to the normal distribution. Some authors however refer the excess kurtosis as kurtosis.

Eg 3.5. Continuation of Eg. 3.4. The kurtosis of a Binomial $Y \sim Bin(n,p)$ is

$$\text{Kur}^{Bin} = 3 + \frac{1-6p(1-p)}{np(1-p)}.$$

For any fixed p, the kurtosis of a Binomial approaches 3 as the sample size $n \to \infty$. When $n = 1$ and $p = 1/2$, Y is a Bernoulli random variable, its Kurtosis is 1, the smallest possible value of the kurtosis.

Distributions that have only location and scale parameters have constant skewness and kurtosis. For example, a normal distribution has skewness 0 and kurtosis 3, exponential 2 and 9 and double exponential 0 and 6.

Estimation of the skewness and kurtosis of a distribution based on a sample $Y_1, \ldots, Y_n$ from that distribution is simply the third and forth sample moments of Y's. Let $\overline{Y}$ and S be the sample mean and standard deviation, the sample skewness and kurtosis are

$$\hat{\text{Sk}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i-\overline{Y}}{S}\right)^3 \quad \text{and} \quad \hat{\text{Kur}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i-\overline{Y}}{S}\right)^4.$$

The sample skewness and kurtosis are highly sensitive to outliers.

Jarque-Bera Test The Jarque-Bera test of normality compares the sample skewness and kurtosis to 0 and 3, those under normality. The test statistic is

$$JB = n\left\{\frac{\hat{\text{Sk}}^2}{6} + \frac{(\hat{\text{Kur}}-3)^2}{24}\right\}$$

The test statistic has a χ^2_2 distribution under the null hypothesis of normal distribution.

```
Sk.fun <- function(x) { ## function to compute skewness
    mean((x-mean(x))^3/sd(x)^3)
}
Kur.fun <- function(x){ ## function to compute kurtosis
    mean((x-mean(x))^4/sd(x)^4)
}
JB.test <- function(x) { ## Jarque-Bera Test
    JB <-length(x)*(Sk.fun(x)^2/6+(Kur.fun(x)-3)^2/24)
    list(stat = JB, pvalue = 1-pchisq(JB,2))
 }
```

Test these R functions with a sample of 75 χ^2_8 random variables.

```
x = rchisq(75,8)
cat("Skewness:", Sk.fun(x));
## Skewness: 0.9561646

cat("Kurtosis:", Kur.fun(x));
## Kurtosis: 3.138031

cat("Jarque-Bera Test:\n", paste(names(JB.test(x)),
round(unlist(JB.test(x)),4), sep = ":", collapse = "   " ))
## Jarque-Bera Test:
##  stat:11.4877   pvalue:0.0032
```

Shape Parameters

A shape parameter is defined as any parameter that is not changed by location and scales change. That is, the value of a shape parameter for the density $f_Z(z)$ of Z equals the value of a shape parameter for $f_Y(y) = \frac{1}{\theta} f(\{y-\delta\}/\theta)$ of $Y = \delta + \theta Z$ in (3.1). The degrees of freedom of t-distributions and the log-standard deviations of log-normal distributions are shape parameters. Shape parameters are often used to specify the skewness or tail weight of a distribution. *Skewness* and *kurtosis* help characterize the shape of a probability distribution.

Eg 3.6. A t_ν distribution, t with ν degrees of freedom, has finite kth moments for $k < \nu$. The kurtosis of t_ν, $\nu > 4$ is

$$\text{Kur}^t(\nu) = \frac{6}{\nu - 4} + 3\,.$$

The kurtosis of a t-distribution decreases as the degrees of freedom ν increase.

Tails of distributions

Heavy tailed distributions are important in finance as we have seen the return distributions have much heavier (or longer) tails than those of normal distributions.

If X is exponential(λ), then its tail probability,

$$P(X > x) = 1 - F(x) = e^{-\lambda x}\,.$$

If $Z \sim N(0,1)$, then it has the tail probability bound,

$$P(Z > z) = 1 - F(z) = \bar{F}(z) < 2e^{-z^2/2}, \qquad \forall\, z \geq 0\,.$$

The tail probability of a normal decays faster than that of a exponential distribution.

Heavy tailed distributions A random variable Y is said to be heavy-tailed if its CDF decays slower than the exponential distribution. The class of power law distributions are heavy-tailed distribution. A random variable Y is power-law distributed if

$$f(y) \sim Ay^{-(a+1)}, \qquad A, a > 0 \quad \text{as} \quad y \to \infty\,.$$

The tail probability of such a random variable has a power law (or polynomial) decay in the right tail of the distribution,

$$P(Y > y) = 1 - F(Y) \sim Ay^{-a}, \qquad y \to \infty\,.$$

Examples of heavy-tailed distributions include lognormal, Pareto, stable and t-distributions.

The t distributions The student's t-distributions are heavy-tailed. We will start with some definitions. Let $Z \sim N(0,1)$, a standard normal, $W \sim \chi^2_\nu$, a chi-square with ν degrees of freedom. If Z and W are *independent*, then the distribution of

$$t_\nu = \frac{Z}{\sqrt{W/\nu}}$$

is called the t distribution with ν degrees of freedom.

If Y has the t_ν distribution then its density,

$$f_{t,\nu}(y) = C(\nu) \cdot \frac{1}{(1+y^2/\nu)^{(\nu+1)/2}} \propto |y|^{-(\nu+1)}, \qquad |y| \to \infty .$$

Thus its tailed probability,

$$P(Y > y) = 1 - F(y) \propto y^{-\nu}, \qquad y \to \infty .$$

The t-distribution has finite kth moments for $k < \nu$. The first two central moments if exist are

$$E(Y) = 0, \quad \nu > 1 \quad \text{and} \quad \text{var}(Y) = \frac{\nu}{\nu-2}, \quad \nu > 2 .$$

The Cauchy distribution is t_1, its first moment mean does not exist. The transformation $\mu + \lambda Y$ is said to have a $t_\nu(\mu, \lambda^2)$ distribution, which has a mean μ and variance $\lambda^2\nu/(\nu-2)$. We will refer these distribution the classical t distributions to distinguish it from the standardized t-distribution that are used in some softwares.

Standardized t-Distributions Some *R* packages use a "standardized" version of the t-distribution. The difference between the two

versions is merely notational, but it is important to be aware of this difference.

For $\nu > 2$, the standardized t is the scale transformation of a t_ν random variable,

$$t^{\text{std}}_\nu = \sqrt{\frac{\nu-2}{\nu}} \cdot t_\nu \sim t_\nu\left(0, \frac{\nu-2}{\nu}\right)$$

so that its mean and standard deviation are 0 and 1. The notation t^{std}_ν means standardized t distribution. More generally, for $\nu > 2$, define the $t^{\text{std}}_\nu(\mu, \sigma^2)$ as

$$t^{\text{std}}_\nu(\mu, \sigma^2) \overset{D}{=} t_\nu\Big(\mu, \frac{\nu-2}{\nu}\sigma^2\Big).$$

The advantage of using the standardized t distribution is that σ^2 is variance while λ^2 of a classical t distribution is not the variance but instead the variance times $(\nu-2)/\nu$.

Generalized error distributions The generalized error distributions add a shape parameter ν to the normal distribution and include the normal distribution. The standardized GED has PDF,

$$f^{\text{STD}}_{\text{GED}}(y;\nu) = \kappa(\nu) \exp\left\{-\frac{1}{2}\frac{|y|^\nu}{\lambda_\nu}\right\}, \qquad y \in \mathbb{R}$$

where $\kappa(\nu)$ and λ_ν are constants given by

$$\lambda_\nu = \sqrt{\frac{\Gamma(1/\nu)}{2^{2/\nu}\Gamma(3/\nu)}} \quad \text{and} \quad \kappa(\nu) = \frac{\nu}{2^{1+1/\nu}\lambda_\nu\Gamma(1/\nu)} .$$

A GED is the standard normal, when $\nu = 2$ and double exponential when $\nu = 1$. The tail probability of a GED,

$$1 - F(y) \propto e^{-y^{\nu}}, \qquad y \to \infty .$$

The GED has slower decay tail probability than a normal if $\nu < 2$ and is heavy tailed $\nu < 1$.

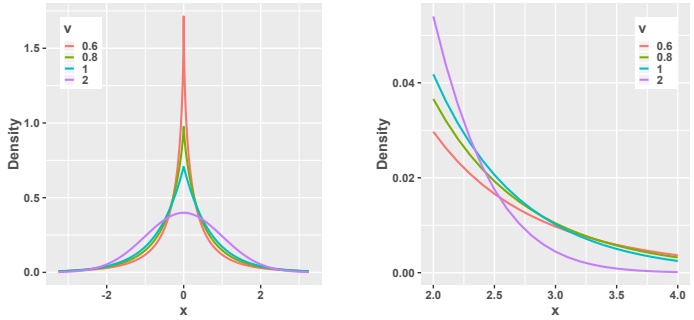

Figure 3.1: The PDFs of GED with the shape parameters, 0.6, 0.8,1 and 2.

The t-distributions and generalized error densities also differ in their shapes at the median. A t-density is smooth and rounded near the median, even with degrees of freedom are small.

Creating skewed from symmetric distributions

In Handout 2, the NASDAQ return data also show skewness in distribution, see Figure 2.3 in addition to heavy tails. Many well-known skewed distributions such as lognormal and gamma distributions are for nonnegative random variables thus cannot be applied to stock return data. The next two classes of skewed distributions are constructed based on a symmetric distribution.

The F-S skewed distributions by Fernandez and Steel (1988). Let f be a symmetric PDF about 0 and ξ a positive constant. Defined

$$f^{\dagger}(y|\xi) = \frac{2}{\xi + \xi^{-1}} f^{*}(y|\xi), \text{ where } f^{*}(y|\xi) = \begin{cases} f(y\xi) & , y < 0 \\ f(y/\xi) & , y \geq 0 \end{cases} .$$

The final version of the skewed distribution is obtained by shifting the location and rescaling such that it has a mean 0 and variance 1. The skewed distributions constructed this way is called the Fernandez-Steel distributions. An F-S distribution with $\xi \neq 1$ is skewed, it is right skewed if $\xi > 1$ and left skewed if $\xi < 1$.

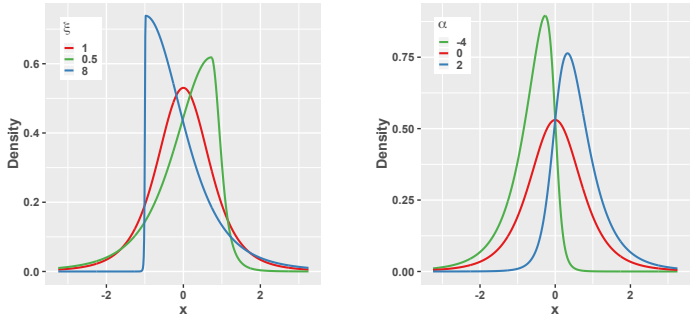

Figure 3.2: The left plot is the PDFs of Fernandez-Steel skewed-t distributions. The right plot is the PDFs of Azzalini and Capitanio's skewed-t distributions. All distributions have degrees of freedom $\nu = 4$.

The A-C skewed distributions Azzalini and Capitanio (2003) have proposed different way to construct skewed normal and t-distributions. These distributions have a shape parameter α that determines the skewness. The A-C skewed normal density is

$$g(y|\xi, \omega, \alpha) = \frac{2}{\omega} \phi(z)\Phi(\alpha z), \qquad z = \frac{y - \xi}{\omega},$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the PDF and CDF of the $N(0,1)$. The parameters ξ, ω and α are called the direct parameters, they determine location, scale and skewness of $g(y)$ but do not have simple interpretations for $g(y)$.

The A-C skew-t distributions have four direct parameters with the forth directed parameter the being degrees of freedom. The A-C skewed distributions can be extended to multivariate distributions. The skewness is positive if $\alpha > 0$ and negative if $\alpha < 0$.

Figure 3.2 shows the F-S and A-C skewed t-densities. The F-S skewed t and GED distributions are included in several *R* packages designed for econometric and financial data, see the Appendix.

Mixture Models

Another class models containing heavy tailed and/or skewed distributions is the set of mixture models.

Discrete Mixtures A distribution f is the PDF of a discrete mixture model if

$$f(x) = \sum_{i=1}^{k} c_i f_i(x)$$

where $f_i(\cdot)$'s are PDFs and the mixing weights, $c_i > 0$, $\sum_i c_i = 1$.

Eg 3.7. A mixture of normals. Let $f(x;\mu,\sigma^2)$, be the PDF of $N(\mu,\sigma^2)$. The simplest example is a random variable Y having the PDF,

$$f(y) = cf(y;\mu_1,\sigma_1^2) + (1-c)f(y;\mu_2,\sigma_2^2), \qquad 0 < c < 1. \quad (3.2)$$

Model (3.2) can be a model of fatter tails and skewed distribution.

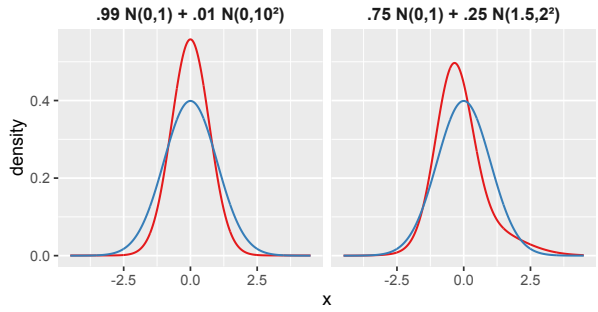

Figure 3.3: Mixtures of normals, all density are standardized .

It is a variance mixture model if $\mu_1 = \mu_2 = 0$ in (3.2), the kurtosis

$$\text{Kur}_Y = \frac{E(Y^4)}{E^2(Y^2)} = \frac{3\{c\sigma_1^4 + (1-c)\sigma_2^4\}}{\{c\sigma_1^2 + (1-c)\sigma_2^2\}^2}.$$

The left plot of 3.3 is with $\sigma_1 = 1, \sigma_2 = 10$ and $c = 0.99$, its kurtosis is 76.5056. A mixing weight $c = 0.95$ gives $\text{Kur}_Y = 42.45$. Higher kurtosis usually implies a longer tailed distribution.

Continuous mixtures The general definition of a normal mixture is the distribution of the random variable

$$Y = \mu + \beta U + \lambda\sqrt{U}Z \quad (3.3)$$

where μ, β, λ are real values with $\lambda > 0$, U is a positive random variable independent of $Z \sim N(0,1)$.

Eg 3.8. *Normal Inverse Gaussian (NIG) distribution*. The NIG distribution, introduced (Bardorff-Nielsen, 1993) specifically for modeling log-return series, is (3.3) with $\lambda = 1$ and $U \sim IG(\delta,\gamma)$, an inverse Gaussian random variable, where $\gamma = \sqrt{\alpha^2 - \beta^2}$. The NIG

is a 4 parameter random variable, $Y \sim NIG(\alpha, \beta, \mu, \delta)$, $\alpha > |\beta|$ and $\delta > 0$. Its tail probability decays as

$$f(y) \sim |y|^{-3/2} \exp(\beta y - \alpha|y|) .$$

The distribution is symmetric around μ if $\beta = 0$, the first 4 central moments of the NIG are

$$E(Y) = \mu + \frac{\delta\beta}{\gamma}, \quad \text{var}(Y) = \frac{\delta\alpha^2}{\gamma^3},$$
$$\text{skewness} = \frac{3\beta}{\alpha\sqrt{\delta\gamma}}, \quad \text{Kurtosis} = \frac{3(1+4(\beta/\alpha)^2)}{\delta\gamma}.$$

Johnson System of Distributions

Johnson (1949) developed a system to continuously transform a normal random variable with a monotone function into random variables having skewed and leptokurtic distributions, e.g. the log-normal distributions. The PDF and CDF of a Johnson system member can be conveniently derived from those of a normal distribution.

The Johnson S_U distribution has been successfully applied to financial asset data. The subscript "U" stands for *unbounded* as a JS_U random variable admits an unbounded support $(-\infty, \infty)$. Let $Z \sim N(0,1)$, the Johnson S_U random variable has 4-parameters in its distribution, $Y \sim JS_U(\gamma, \delta, \xi, \lambda)$, $\delta > 0, \lambda > 0$, such that,

$$Z = \gamma + \delta \sinh^{-1}\left(\frac{Y-\xi}{\lambda}\right)$$

where the inverse sinh function $\sinh^{-1}(u) = \log(u + \sqrt{(u^2+1)})$ cuz $\sinh(u) = (e^u - e^{-u})/2)$. Immediately, its CDF is given by

$$F(y) = \Phi\left\{\gamma + \delta \sinh^{-1}\left(\frac{y-\xi}{\lambda}\right)\right\}$$

and the PDF is the derivative,

$$f(y) = \frac{\delta}{\lambda\sqrt{2\pi}\sqrt{1+\left(\frac{y-\xi}{\lambda}\right)^2}}\, \phi\left\{\gamma + \delta \sinh^{-1}\left(\frac{y-\xi}{\lambda}\right)\right\},$$

where ϕ and Φ are the PDF and CDF of $N(0,1)$. The quantile function of Y simply depends on $\Phi(\cdot)$, that is,

$$F^{-1}(q) = \xi + \lambda \sinh\left\{\frac{\Phi^{-1}(q) - \gamma}{\delta}\right\}.$$

The JS_U has been applied often to VaR in risk management for the simplicity of its quantile function. Let $\omega = \exp(-\delta^2)$ and $\Omega = \gamma/\delta$, the mean and variance of JS_u are

$$E[Y] = \xi - \lambda\omega^{1/2}\sinh\Omega$$
$$\text{var}(Y) = \frac{1}{2}\lambda^2(\omega - 1)(\omega\cosh 2\Omega + 1)$$

where $\cosh(u) = (e^u + e^{-u})/2$. Both skewness and Kurtosis of S_U distribution can be expressed explicitly with ω and Ω.

Maximum Likelihood Estimation

The most common method for estimating parameters in a parametric model is the maximum likelihood method.

Likelihood Function

Let $f(\boldsymbol{Y}|\boldsymbol{\theta})$ be the PDF of the observed data $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T$, where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)^T$ is a vector of parameters. The likelihood function $L(\boldsymbol{\theta}) = f(\boldsymbol{Y}|\boldsymbol{\theta})$ which is viewed as a function of $\boldsymbol{\theta}$ with $\boldsymbol{Y}$ fixed at the observed data. If $Y_1, \ldots, Y_n$ are $i.i.d.$ with PDF $f(y|\boldsymbol{\theta})$, then the likelihood and the log likelihood functions are

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} f(y_i \,|\, \boldsymbol{\theta}) \quad \text{and} \quad \log L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log f(y_i \,|\, \boldsymbol{\theta}). \qquad (3.4)$$

Mathematically, it is easier to maximize the log-likelihood, $\log\{L(\boldsymbol{\theta})\}$.

Eg 3.9. Normal distributions. For $Y_i \sim N(\mu, \sigma^2)$, $\boldsymbol{\theta} = (\mu, \sigma^2)'$,

$$L\{(\mu, \sigma^2)'\} = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \mu)^2 \right\},$$

$$\log L((\mu, \sigma^2)') = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - \mu)^2 .$$

In R, the likelihood and log-likelihood functions for the distributions we have discussed can easily be written with the "d" functions.

```
args(dnorm) ## see the arguments and default settings
## function (x, mean = 0, sd = 1, log = FALSE)

## observed data: x
lik = function(theta) prod( dnorm(x, theta[1], theta[2]))
loglik = function(theta) sum(dnorm(x, theta[1], theta[2], log = T))
```


Maximum Likelihood Estimator

The MLE, is the value of $\boldsymbol{\theta}$ that maximizes $L(\boldsymbol{\theta})$. Maximizing either the log-likelihood or likelihood will lead to the same answer. We denote the MLE by $\hat{\boldsymbol{\theta}}_{\text{ML}}$. If the likelihood function is differentiable in θ_i, possible candidates for the MLE are the values of $(\theta_1, \ldots, \theta_k)'$ that solve

$$\frac{\partial}{\partial \theta_i} L(\boldsymbol{\theta} \,|\, \boldsymbol{y}) = 0, \qquad i = 1, \ldots, k.$$

Note that the solutions to the above equation are merely possible candidates for the MLE. Points at which the first derivatives are 0 may be local or global maxima or inflection point.

Eg 3.10. Eg. 3.9 continued.

$$\frac{\partial}{\partial \mu} \log L = \frac{1}{\sigma^2} \sum_{i=1}^{n} (Y_i - \mu) \overset{\text{set}}{=} 0 \implies \hat{\mu} = \overline{Y},$$

$$\frac{\partial}{\partial \sigma^2} \log L = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4} \sum_{i=1}^{n} (Y_i - \mu)^2 \implies \hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \overline{Y})^2 .$$

Often in practice, the MLE cannot be solved analytically and requires numerical iterative methods.

Fisher Information and the Observed Fisher Information

Standard errors are essential to any estimators for statistical inference. For the MLE, the standard error is obtained from the Fisher informations matrix of the sample $\mathcal{I}(\boldsymbol{\theta})$ defined as

$$\mathcal{I}(\boldsymbol{\theta}) = -E\left\{ \frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'} \log L(\boldsymbol{\theta}) \right\}.$$

Result 3.1. *Under suitable assumptions, for large enough sample sizes, the maximum likelihood estimator is approximately normally distributed with mean equal to the true parameter and with variance equal to the inverse of the Fisher information matrix at* $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$, $\mathcal{I}^{-1}(\hat{\boldsymbol{\theta}})$.

If θ is one-dimensional, then then the Fisher information is simply $\mathcal{I}(\theta) = -E\{\partial^2 \log L(\theta)/\partial\theta^2\}$, the standard error of $\hat{\theta}$ is given by ,

$$s_{\hat{\theta}} = \left\{\mathcal{I}(\hat{\theta})\right\}^{-1/2},$$

and the Central Theorem of Result 3.1 gives the $100(1-\alpha)\%$ confidence interval for θ ,

$$\hat{\theta} \pm z_{\alpha/2} s_{\hat{\theta}}, \tag{3.5}$$

where $z_{\alpha/2}$ is the $\alpha/2$-upper quantile of the standard normal.

It is not always possible to calculate the Fisher information as it is and expectation. The *observed Fisher information* is the Fisher information without the expectation,

$$\mathcal{I}^{\text{obs}}(\hat{\boldsymbol{\theta}}) = -\frac{\partial^2}{\partial\boldsymbol{\theta}\,\partial\boldsymbol{\theta}'} \log L(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}.$$

Often the observed information is used in place of the fisher information in constructing confidence intervals. There is theory suggesting that using the observed Fisher information will result in a more accurate confidence interval.

Eg 3.11. Exponential distribution, see also Eg. 3.3. Suppose $Y_i \sim Exp(\theta)$ with a PDF $f(\theta) = \frac{1}{\theta}\exp\left\{-\frac{y}{\theta}\right\}$, see page 33. Then the

log-likelihood $\log L(\theta) = -n\log\theta - \frac{1}{\theta}\sum_{i=1}^n Y_i$.

$$\frac{\partial \log L}{\partial\theta} = -\frac{n}{\theta} + \frac{1}{\theta^2}\sum_{i=1}^n Y_i \implies \hat{\theta} = \overline{Y},$$

$$\frac{\partial \log L}{\partial\theta^2} = \frac{n}{\theta^2} - \frac{2}{\theta^3}\sum_{i=1}^n Y_i \implies \mathcal{I}(\hat{\theta}) = E\Big\{-\frac{n}{\overline{Y}^2} + \frac{2}{\overline{Y}^3}\sum_{i=1}^n Y_i\Big\} = E\left\{\frac{n}{\overline{Y}^2}\right\}.$$

The observed Fisher information $\mathcal{I}^{obs}(\hat{\theta}) = n/\overline{Y}^2$, thus, the standard error of $\hat{\theta}$, $s_{\hat{\theta}} = \left\{\mathcal{I}^{obs}(\hat{\theta})\right\}^{-1/2} = \overline{Y}/\sqrt{n}$.

Fitting Distributions in R

The MLE of normal and exponential distributions are two of very few examples that can be solved analytically. In most of cases, the MLE must be obtained numerically in an optimization problem. R's `optim()`, `nlminb()` and `nlm()` are functions for minimizing an objective function and can be applied to $-L(\boldsymbol{\theta})$ or $-\log L(\boldsymbol{\theta})$.

- `optim()` – has 4 numerical algorithm options. It allows box-constraints and returns Hessian matrix as an option.
- `nlm()` – uses Newton-type algorithm. No constraint can be specified. It returns Hessian matrix as an option.
- `nlminb()` – is similar to `nlm()` but allows box-constraints. It does not return Hessian matrix.

We will show various ways to fit distributions in R:

1. Minimize a negative log-likelihood $-\log L(\boldsymbol{\theta})$ with an R function.

2. Fit distributions with R's `fitdistr()`. The function uses `optim()` for its optimization.
3. Fit distribution with R functions of R packages, `fGarch` and `rugarch` for distributions such as skewed distributions that are not in R.

The data set is the weekly NASDAQ returns from Handout 2.

```
x = weeklyReturn(IXIC, type = "log")
x = x*100 ## convert return to %
head(x, 1); tail(x,1);n = dim(x)[1];cat("n =",n)
##            weekly.returns
## 1984-01-06      3.283609
##            weekly.returns
## 2024-08-16      5.158192
## n = 2120
```

Optimizing likelihood in R. If we would like to be able to set lower or upper bounds of our estimates or/and calculate standard errors, `optim()` is the most suitable choice.

Eg 3.12. Fit the t distribution with R's `optim()`. The first step is to write the log-likelihood function (3.4). The `dt()` in R is for the PDF of standard t_ν, it has only degrees of freedom as its sole parameter. We can write the PDF for $Y_i \sim t_\nu(\mu, \lambda^2) = \mu + \lambda t_\nu$ with the formula (3.1). Let f_ν denote the PDF of t_ν, then $\boldsymbol{\theta} = (\mu, \lambda, \nu)'$ and the log-likelihood of $(Y_1, \ldots, Y_n)'$ is

$$\log L(\boldsymbol{\theta}) = \log\left\{\prod_{i=1}^{n} \frac{1}{\lambda} f_\nu\Big(\frac{y_i - \mu}{\lambda}\Big)\right\} = \sum_{i=1}^{n}\left\{\log f_\nu\Big(\frac{y_i - \mu}{\lambda}\Big) - \log\lambda\right\}.$$


```
loglik_t = function(theta){## negaive log-likelihood
    -sum(dt((x-theta[1])/theta[2], df = theta[3], log = T) - log(theta[2]))
}
```

Next, we use R's `optim()` function to minimize the negative log-likelihood `loglik_t`.

```
args(optim)
## function (par, fn, gr = NULL, ..., method = c("Nelder-Mead",
##     "BFGS", "CG", "L-BFGS-B", "SANN", "Brent"), lower = -Inf,
##     upper = Inf, control = list(), hessian = FALSE)
```

The first two arguments are necessary, they are the starting values and the objective function to be optimized. Both scale (λ) and DF (ν) are positive, the lower bound `lower.` needs to be reset. This also requires to change the optimization method to `"L-BFGS-B"`. Also, by setting `hessian = T`, the observed Fisher information matrix will be computed numerically and returned.

```
start = c(m = mean(x), s = sd(x), df = 4)
fit_t1 = optim(start, loglik_t, hessian = T, method = "L-BFGS-B",
               lower = c(-100, 0.0001, 0.1));
names(fit_t1);
## [1] "par"         "value"       "counts"      "convergence" "message"
## [6] "hessian"

fit_t1$par ## Estimates
##         m         s        df
## 0.3518153 2.0086319 3.6482708

fit_t1$value ## negative log-likelihood at MLE
## [1] 5099.903

fit_t1$hessian ## Hessian matrix
##             m         s         df
## m  368.303962  16.51369   4.546954
## s   16.513689 573.32855 -69.239610
## df   4.546954 -69.23961  18.532776
```

Note: (1) the label `s` stands for scale which is denoted by λ. (2) Our objective function `loglik_t()` is the negative log-likelihood.

The covariance matrix is the inverse of the Hessian, which can be computed by R's `solve()`. The standard errors is simply the square root of the diagonal elements of the covariance matrix, the relative R functions are `sqrt()` and `diag()`.

```
solve(fit_t1$hessian) ## inverting Hessian - cov of estimates
##                 m             s           df
## m   0.0027501134 -0.0002928145 -0.001768704
## s  -0.0002928145  0.0032093615  0.012062216
## df -0.0017687043  0.0120622161  0.099457595

se_t = sqrt(diag(solve(fit_t1$hessian))) ## compute Std Err
cat("Standard Error of Estimates\n"); se_t

## Standard Error of Estimates
##          m          s         df
## 0.05244152 0.05665123 0.31536898

par = fit_t1$par
sd = par["s"]*sqrt(par["df"]/(par["df"]-2)) ## sigma estimate
sd.se = se_t["s"]*sqrt(par["df"]/(par["df"]-2)) ## Std Err of sd
cat("sigma Estimate:", sd, "\tse;", sd.se)

## sigma Estimate: 2.988337  se; 0.08428272
```

The last computation is computing the standard deviation estimate. Recall that the scale parameter λ of a classical t distribution is not standard deviation, instead, $\lambda^2 = \sigma^2(\nu-2)/\nu$,see pages 38-40.

Fitting distributions with R's `fitdistr()`. If the target distribution is one of R's probability distributions or the density function is available, the simplest way to obtain the MLE and its standard errors is applying `fitdistr()`.


```
library(MASS)
args(fitdistr)

## function (x, densfun, start, ...)
```

The package `MASS` is installed with R, we only need to load it. It includes many useful functions and data examples. The `densfun` argument is either a density R function, e.g. `dnorm`, or a distribution name that is recognized by R, e.g. `"norm"` or `"t"`. The `start` argument is required and must be in a list vector.

Eg 3.13. Fit t distribution with R's `fitdistr()`. This is the same fit as in Eg. 3.12 without having to calculating the standard errors.

```
start = list(m = mean(x), s = sd(x), df =  4)
fit_t2 = fitdistr(x, "t", start = start, lower = c(-100, 0.0001, 0.1))
fit_t2
##        m             s             df
##   0.35181534   2.00863187   3.64827084
##  (0.05244152) (0.05665123) (0.31536898)

names(fit_t2) ## return value

## [1] "estimate" "sd"       "vcov"     "loglik"   "n"

fit_t2$vcov ## covariance matrix of estimates

##                 m             s           df
## m   0.0027501134 -0.0002928145 -0.001768704
## s  -0.0002928145  0.0032093615  0.012062216
## df -0.0017687043  0.0120622161  0.099457595

fit_t2$loglik ## log-likelihood at MLE

## [1] -5099.903
```

The return values include the MLE (`estimate`), standard errors

of the MLE (sd), covariance matrix of the MLE (vcov), the log-likelihood evaluated at MLE ((loglik)) and the sample size ((n)).

Eg 3.14. Fit t^{std} and skewed t^{std} distributions with R's `fitdistr()`. The `fGarch` package includes distribution functions of the standardized t (std), GED (ged),and their F-S skewed distributions (sstd andsged). See the details in Appendix. In this example, we will use the density functions `dstd()` and `dsstd()`.

```
library(fGarch)
args(dstd)
## function (x, mean = 0, sd = 1, nu = 5, log = FALSE)

start = list(mean = mean(x), sd = sd(x), nu = 4)
fit_std = fitdistr(x, dstd, start = start, lower = c(-100, 0.0001, 2.01))
fit_std

##       mean          sd          nu
##   0.35180703   2.98850818   3.64787579
##  (0.05244151) (0.11762026) (0.31527348)
```

The `start` vector should be setup with labels (`mu`, `sd`, `nu`) the same as the arguments of the density function specified. This fit is essentially the same as those in Egs. 3.12 and 3.13, the estimates of mean and DF only differ slightly numerically, `sd` is the estimate of σ (not λ). Next output we fit the F-S skewed standardized t distribution.

```
args(dsstd)
## function (x, mean = 0, sd = 1, nu = 5, xi = 1.5, log = FALSE)

start = list(mean = mean(x), sd = sd(x), nu = 4, xi = 1)
fit_sstd = fitdistr(x, dsstd, start = start, lower = c(-100, 0.0001, 2.01, 0.01))
fit_sstd
##       mean          sd          nu          xi
##   0.20633595   2.97875698   3.74122044   0.87893227
##  (0.06206621) (0.11447485) (0.32811305) (0.02600179)
```


The confidence intervals of any parameter can construct easily with the formula (3.5). Suppose the skewed t distribution is our chosen model, the CIs are computed below.

```
## Compute 95% CI for mean
fit_sstd$est["mean"] + c(-1,1)*qnorm(.975)*fit_sstd$sd["mean"]
## [1] 0.08468842 0.32798348

## Compute 95% CI for all parameters
est = fit_sstd$est; se = fit_sstd$sd
ci = cbind(est, est - qnorm(.975)*se, est + qnorm(.975)*se)
colnames(ci) = c("est", "lower.95%", "upper.95%"); ci

##            est  lower.95% upper.95%
## mean 0.2063359 0.08468842 0.3279835
## sd   2.9787570 2.75439040 3.2031236
## nu   3.7412204 3.09813069 4.3843102
## xi   0.8789323 0.82796970 0.9298948
```

Fitting distributions with R packages. The package `fGarch` also have R functions for fitting distributions, e.g. `stdFit()` for fitting a standard t distribution, `sstdFit()` for skewed standard t. These functions only calculate the MLE's without standard errors or Hessian matrix and their help documentation files are not concise. We will skip fitting distribution with `fGarch` R functions.

We will focus on the `rugarch` package, which will be used later for modeling GARCH models. There are eight univariate distributions in `rugarch` including normal inverse Gaussian and Johnson S_U distributions. The setup of the distribution functions are uniform among the eight (see Appendix) with the `mu` and `sigma` for μ and σ. We can fit these distributions of with its fitting R function `fitdist()`.

Eg 3.15. Fit distributions with `rugarch`'s `fitdist()`. We will demonstrate for standardized t and its F-S skewed version as Eg. 3.14.

```
library(rugarch)
args(fitdist)
```

```
## function (distribution = "norm", x, control = list())
```

```
fit_std2 = fitdist("std", x) ## fit x with std
names(fit_std2) ## return value names
```

```
##  [1] "pars"        "convergence" "values"      "lagrange"    "hessian"
##  [6] "ineqx0"      "nfuneval"    "outer.iter"  "elapsed"     "vscale"
```

```
fit_std2$pars  ## estimates
```

```
##        mu     sigma     shape
## 0.3518092 2.9884995 3.6479159
```

```
fit_std2$values ## a vector of negative log-lik values
```

```
## [1] 5104.589 5099.903 5099.903
```

```
tail(fit_std2$values, 1)  ## neagtive log-likelihood at MLE
```

```
## [1] 5099.903
```

The `value` in the output is a vector of negative log likelihood values from major iterations, only the last one is the solution and it is the negative log-likelihood evaluated at the estimates. The standard errors are required to be computed from the Hessian.

```
std2_se = sqrt(diag(solve(fit_std2$hessian))) ## compute Std Err of estimates
names(std2_se) = names(fit_std2$pars); std2_se
```

```
##        mu     sigma     shape
## 0.05234104 0.11722928 0.30999102
```

Next we fit the F-S skewed t distribution with `rugarch`'s `fitdist()`.


```
fit_sstd2 = fitdist("sstd", x) ## fit x with std
rbind(estimates =fit_sstd2$pars,
      std.err = sqrt(diag(solve(fit_sstd2$hessian))))
```

```
##                    mu     sigma       skew     shape
## estimates 0.20633067 2.9787656 0.87893270 3.7411912
## std.err   0.06184895 0.1134682 0.02640203 0.3531641
```

We show in Egs. 3.14 and 3.15 to use `MASS`'s `fitdistr()` and `rugarch`'s `fitdist()` to fit distributions. The latter only works within the `rugarch` package, the former can be applied to any PDF in simple form of `R` function (not `rugarch`'s ddist()). Furthermore, `fitdistr()` can fit a distribution with one or more parameters hold fixed. This is required to calculating a likelihood ratio test statistic.

Likelihood Ratio Tests

Suppose that $\boldsymbol{\theta}$ is a length-p parameter vector and the null hypothesis puts $m \le p-1$ constraints on $\boldsymbol{\theta}$. That is, there are m function $g_1, g_2, \ldots, g_m$ and the null hypothesis is that $H_0 : g_i(\boldsymbol{\theta}) = 0$ for $i = 1, \ldots, m$. The models without and with the constraints are called the full and reduced (or restricted) models, respectively.

Let $\hat{\boldsymbol{\theta}}_{\text{ML}}$ be the MLE based on the full model and $\hat{\boldsymbol{\theta}}_{0,\text{ML}}$ be the MLE based on the reduced model subject to the constraints of H_0. If H_0 is true, $\hat{\boldsymbol{\theta}}_{\text{ML}}$ and $\hat{\boldsymbol{\theta}}_{0,\text{ML}}$ would be both close to the true $\boldsymbol{\theta}$, the difference between them would be small. If H_0 is false, then $\hat{\boldsymbol{\theta}}_{0,\text{ML}}$ is far from $\hat{\boldsymbol{\theta}}$ so that $L(\hat{\boldsymbol{\theta}}_{0,\text{ML}})$ is much smaller than $L(\hat{\boldsymbol{\theta}}_{\text{ML}})$. The likelihood ratio test rejects H_0 if

$$2\Big\{\log L(\hat{\boldsymbol{\theta}}_{\text{ML}}) - \log L(\hat{\boldsymbol{\theta}}_{0,\text{ML}})\Big\} \ge c\,,$$

where c is a critical value. A critical value is exact if it gives a level that is exactly equal to α. When an exact critical value is unknown, then the usual choice of the critical value is

$$c = \chi^2_{\alpha,m}\,,$$

the α-upper quantile value of the chi-squired distribution with m degrees of freedom. This is based on the large sample theory that asymptotically, the LRT statistic has a chi-squared distribution.

Eg 3.16. The normal distribution, Eg. 3.10 continued. Suppose that we are testing $H_0 : \mu = 0$. The likelihood under H_0 is

$$\log L_0(\sigma^2) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{n} Y_i^2$$

$$\Longrightarrow \hat\sigma_0^2 = \frac{1}{n}\sum_{i=1}^{n} Y_i^2 \Longrightarrow \log L_0(\hat\sigma_0^2) = -\frac{n}{2}\log\hat\sigma_0^2 - \frac{n}{2}\log(2\pi) - \frac{n}{2}$$

For the full model, $\hat\mu = \overline{Y}$ and $\hat\sigma^2 = n^{-1}\sum_i (Y_i - \overline{Y})^2$, and

$$\log L(\hat\mu, \hat\sigma^2) = -\frac{n}{2}\log\hat\sigma^2 - \frac{n}{2}\log(2\pi) - \frac{n}{2}$$

The likelihood ratio statistic is

$$\text{LRT} = n\{-\log\hat\sigma^2 + \log\hat\sigma_0^2\}\,.$$

Eg 3.17. Compute LRT with `fitdistr()`. Suppose the full model is the skewed t distribution fitted in Eg 3.14 (`fit_sstd`). We want to test if the average weekly return is 0, $H_0 : \mu = 0$. The likelihood function under H_0, by setting `mean = 0`, has 3 parameters instead

of 4. A reduced model can easily be fit with `fitdistr()`.

```
start0 = list(sd = sd(x), nu = 4, xi = 1)  ## 3 parameters without mean
fit_sstd0 = fitdistr(x, dsstd, start = start0, mean = 0,
                     lower = c(0.0001,  2.01, 0.01))
fit_sstd0
```

```
##        sd           nu           xi
##   3.09611249   3.62604644   0.83522997
##  (0.13024045) (0.32531963) (0.02094762)
```

```
cat("log-likelihoods:\n");c(full = fit_sstd$loglik, reduced = fit_sstd0$loglik)
```

```
## log-likelihoods:
##      full   reduced
## -5090.251 -5095.567
```

```
LRT = 2*(fit_sstd$loglik-fit_sstd0$loglik)  ## Computing LRT
c(LRT.statistic = LRT, p.value = 1-pchisq(LRT,1))
```

```
## LRT.statistic       p.value
##  10.632837655   0.001110971
```

The p-value 0.00111 is small, we can reject H_0. Suppose we would like to test $H_0 : \mu = 0$ & df $= 4$, then the reduced model has 2 parameters σ and ξ.

```
start0 = list(sd = sd(x), xi = 1)
fit_sstd0 = fitdistr(x, dsstd, start = start0, mean = 0, nu = 4,
                     lower = c(0.0001, 0.01))
cat("log-likelihoods:\n");c(full = fit_sstd$loglik, reduced = fit_sstd0$loglik)
```

```
## log-likelihoods:
##      full   reduced
## -5090.251 -5096.136
```

```
LRT = 2*(fit_sstd$loglik-fit_sstd0$loglik)  ## Computing LRT
cat("LRT:", LRT, "p-value" , 1-pchisq(LRT,2))
```

```
## LRT: 11.77124 p-value 0.002779122
```

The test is again significant, we can reject H_0, at least of of the equality in H_0 does not hold.

Model Selection and Information Criteria

The maximized value of the log-likelihood, denoted by $\log L(\hat{\boldsymbol{\theta}}_{\text{ML}})$, can be used to measure how well a model fits the data or to compare the fits of two or more models. However, $\log L(\hat{\boldsymbol{\theta}}_{\text{ML}})$ can be increased simply by adding parameters to the model. To find a parsimonious model one needs a good tradeoff between maximizing fit and minimizing model complexity.

AIC (Akaike's information criterion) and BIC (Bayesian information criterion) are two means for this purpose. They are defined by

$$\begin{aligned}\text{AIC} &= -2\log L(\hat{\boldsymbol{\theta}}_{\text{ML}}) + 2p \\ \text{BIC} &= -2\log L(\hat{\boldsymbol{\theta}}_{\text{ML}}) + p\log n\,.\end{aligned}$$

where p is the number of parameters in the model. The terms $2p$ and $p\log n$ are called "complexity penalties" since they penalize larger models.

The minus twice log-likelihood term $-2\log L(\hat{\boldsymbol{\theta}}_{\text{ML}})$ is called deviance, so that the information criterion is "deviance + penalty". Deviance quantifies model fit, with smaller values implying better fit.

Generally, from a group of candidate models, one selects the model that minimizes whichever criterion, AIC or BIC, is being used. Since $\log n > 2$ for $n \geq 8$, BIC penalizes model complexity more than AIC does, and thus BIC tends to select simpler models than AIC.

Eg 3.18. Eg. 3.14 continued. In addition to t^{std} and skewed t^{std} in Eg. 3.14. we fit 2 more candidate models, the GED and F-S

skewed GED. The CDF of these two distributions are available in the `fGarch`. As in Eg. 3.14, we will use `fitdistr()`.

```
## **** fit GED **** ##
args(dged)
## function (x, mean = 0, sd = 1, nu = 2, log = FALSE)

start = list(mean = mean(x), sd = sd(x), nu = 1) ## fit skewed GED
fit_ged = fitdistr(x, dged, start = start,lower = c(-100, 0.0001, 0.01))

## **** fit skewed GED **** ##
args(dsged)
## function (x, mean = 0, sd = 1, nu = 2, xi = 1.5, log = FALSE)

start = list(mean = mean(x), sd = sd(x), nu = 1, xi = 1) ## fit skewed GED
fit_sged = fitdistr(x, dsged, start = start,lower = c(-100, 0.0001, 0.01, 0.01))

## **** Compute IC **** ##
dists = c("std", "sstd","ged", "sged") ## distribution names
fits = list(fit_std, fit_sstd, fit_ged, fit_sged) ## put 4 fits together
loglik = c(); p = c() ## specify output
for(i in 1:length(dists)){
        loglik[i] = fits[[i]]$loglik
        p[i] = length(fits[[i]]$est)
}
names(loglik) = names(p) = dists
aic = -2*loglik + 2*p ## Compute AIC
bic = -2*loglik + p*log(n) ## Compute BIC
rbind(loglik, p, aic, bic)

##              std      sstd       ged      sged
## loglik -5099.903 -5090.251 -5114.416 -5105.131
## p          3.000     4.000     3.000     4.000
## aic    10205.806 10188.501 10234.833 10218.262
## bic    10222.783 10211.138 10251.810 10240.899

cat("Model selected by AIC:", dists[which.min(aic)], "\nModel selected by BIC:",
    dists[which.min(bic)])

## Model selected by AIC: sstd
## Model selected by BIC: sstd
```

Both AIC and BIC select skewed t distribution. The estimate of ξ is 0.8789, seen in Egs. 3.14 and 3.15, reflecting the left skewness shown in Figure. 2.3. We next show how to utilize `rugarh`'s `fitdist()` to fit several distributions.

Eg 3.19. Eg. 3.15 continued. Two more models will be included, the normal inverse Gaussian and Johan S_u.

```
dists = c("std", "sstd", "ged", "sged", "nig", "jsu") ## distribution names
fits = vector("list", length = length(dists)); names(fits) = dists
loglik = c(); p = c()
for(i in 1:length(dists)){
        fits[[i]] = fitdist(dists[i], x)
        loglik[i] = -tail(fits[[i]]$values,1)  ## log-likelihood
        p[i] = length(fits[[i]]$pars)
}
names(loglik) = names(p) = dists
aic = -2*loglik + 2*p
bic = -2*loglik + p*log(n)
rbind(loglik, p, aic, bic)
```

```
##              std      sstd       ged     sged       nig       jsu
## loglik -5099.903 -5090.251 -5114.416 -5105.13 -5090.564 -5089.029
## p          3.000     4.000     3.000     4.00     4.000     4.000
## aic    10205.806 10188.501 10234.833 10218.26 10189.127 10186.059
## bic    10222.783 10211.138 10251.810 10240.90 10211.764 10208.695
```

```
cat("Model selected by AIC:", dists[which.min(aic)],
    "\nModel selected by BIC:", dists[which.min(bic)])
```

```
## Model selected by AIC: jsu
## Model selected by BIC: jsu
```

Recall that `rugarch`'s `fitdist()` returns negative log-likelihood values. Both AIC and BIC choose Johnson S_U distribution. It is worth noting that AIC or BIC or any information criterion cannot tell us how well a particular model explains our data, it only tells

us if it is a better model than other models in terms of the balance between model complexity and maximizing fit . We should always check the final model with QQ plots and/or density plots.

Figure 3.4 shows the density plots and QQ plots of Johnson S_U distribution and skewed t distribution, which are the “best” and "2nd best" according to both AIC and BIC. The difference seems to be at the right tail, where Johnson S_U fits slightly better.

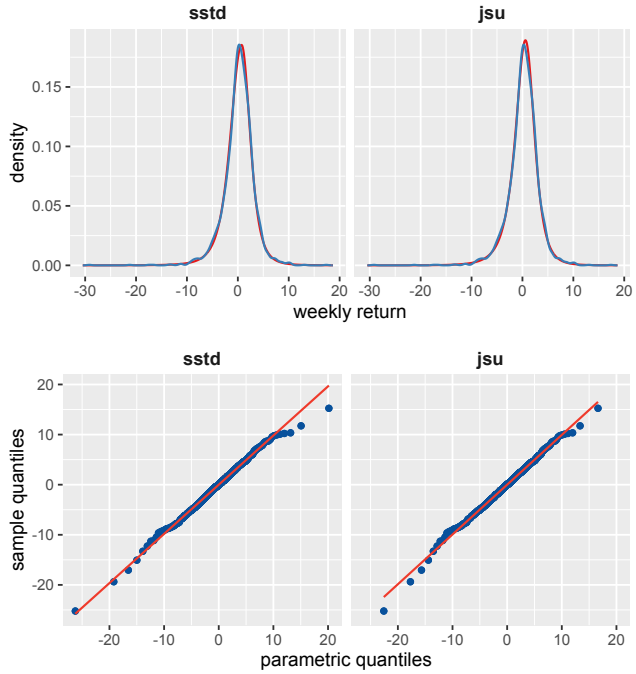

Figure 3.4: The upper panel is the density plots of parametric models in red with the kernel density curve superimposed in blue. The lower panel is the sample quantiles plotted against those of parametric models.

Appendix

This appendix is to give a brief description of probability distributions R's packages `fGarch` and `rugarch`. Both packages include the following distributions:

Distribution	R **Name**	**Distribution**	R **Name**
standardized t	`std`	skewed t	`sstd`
GED	`ged`	skewed GED	`sged`
normal	`norm`	skewed normal	`snorm`

Only in `rugarch` :

Normal Inverse Gaussian	`nig`	Johnson S_U	`jsu`

The three skew distributions are the Fernandez-Steel distributions.

Distribution Functions

`fGarch`. The distribution functions in `fGarch` are just like the regular R distribution functions. Prefix the name by `d` for the density, `p` for the CDF, `q` for the quantile function and `r` for simulation. Eg. `dstd()`, `pstd()`, `qstd()` and `rstd()`.

`rugarch`. The distribution functions in `rugarch` are `ddist()` for the density, `pdist()` for the CDF, `qdist()` for the quantile function and `rdist()` for simulation with the first argument being the distribution name. Eg. `ddist("std", ...)`, `pdist("std", ...)`, `qdist("std", ...)` and `rdist("std", ...)`.

Fitting Distribution

`fGarch`. The function of ML fitting a distribution is suffixing the distribution by `Fit`. Eg. `stdFit()`. All distribution fittings use `nlminb()` to optimize except for the skewed t distribution which uses `nlm()`. Thus, only fitting the skewed t distribution returns the Hessian matrix. All fits return negative log-likelihood evaluated at MLE.

`rugarch`. The function of ML fitting is `fitdist(distribution, ...)`. Eg. `fitdist(distribution = "std",...)`. The optimization uses an optimization function from R package `Rsolnp` called `solnp()`. The function optimizes using augmented Lagrange multiplier method. The return value includes the Hessian matrix and negative log-likelihood evaluated at MLE along with those from major iterations.