Statistical Foundation for Finance: Assignment 1

Christoffer Guttulsrød (23-717-416), Zhichao Pan (23-746-993), and Oliver Löthgren (23-702-780)

University of Zurich

8th November, 2023

Abstract

The Kolmogorov-Smirnov (KD) and Anderson-Darling (AD) are two widely used goodness-of-fit test. This paper studies and compares the power of these two tests for unspecified location-scale t-distribution. The critical values and power estimation were obtained via Monte Carlo simulation of sample data generated from symmetric but non-fat-tailed alternative distributions. Our simulation results show that KD exhibits more consistent behaviour and higher power than AD in this specific case. Furthermore, we explore the inversion theorem of the characteristic function (cf) to obtain the probability density function (pdf) of a location scale t-distribution. The pdfs resulting from the convolution formula, Gaussian kernel density estimate (KDE), and inversion theorem are also compared in the setting of a sum of two independent Student's t-distributed random variables. The inversion approach is found to obtain accurate pdfs relative to the well-defined pdfs. The Gaussian KDE is also found to perform well for large samples with an appropriate tail trimming. Thereafter, expected shortfall (ES) is calculated for t-distributed data, with confidence intervals constructed by use of parametric and non-parametric bootstraps. For smaller sample sizes the parametric bootstrap has an edge over the non-parametric bootstrap both for accuracy and precision. For larger sample sizes the non-parametric method appears to be more stable around the nominal coverage compared to the parametric.

Contents

1 Introduction **3**

2 Goodness of Fit Test with KD and AD **3**
2.1 Simulation Methodology 4
2.2 Results and Discussion 5

3 Obtaining Probability Density Functions **8**
3.1 The Student's t-distribution: Analytic PDF and the inverted CF 8
3.2 PDF of the Sum of two independent Student's t-distributions 10

4 Expected Shortfall **13**
4.1 Methodology 13
4.1.1 True Expected Shortfall 13
4.1.2 The Bootstrap and CI 13
4.1.3 Coverage & CI length 14
4.2 Results and Discussion 14

A Plots **17**
A.1 Convergence of Actual Coverage in Sample Size 17

B Tables **18**

C Python Code **19**
C.1 KD and AD Statistics 19
C.2 Estimation of Cutoff Values 19
C.3 Evaluation of Power 21
C.4 PDF plots of Student's t-distribution 22
C.5 PDF plots for the Sum of two independently distributed Student's t-distributed random variables 22
C.6 True expected shortfall 23
C.7 ES bootstrap CI MLE 24
C.8 Coverage and CI Length 25
C.9 Coverage for multiple sample sizes 26

1 Introduction

The KD and AD test statistics employ the empirical cumulative distribution function (ecdf) to quantify the accuracy of a proposed null distribution (with a well-defined cdf); the ecdf values are compared to those of the analytic null cdf at points corresponding to the data samples. In particular, the KD statistic measures the maximal absolute value difference between the ecdf and null cdf evaluated at the data points. The AD statistic is similar, but includes an additional divisive factor as a function of the cdf that places more emphasis on differences in the tails of the distribution (equations are given in section 2 below). In this study, we explore the composite hypothesis H_0 : the data is an i.i.d sample from a location-scale t-distribution against the alternative H_1 : the data is not drawn from a location-scale t-distribution. To critically assess the efficacy of the KD and AD statistics for this purpose we wish to analyse their respective size and power. We begin by calculating appropriate α level cutoff values under H_0 for the parameterisations $df = 3, 6$, and 12 via simulation. Note that different values for location and scale are not explored as the H_0 distributions of the statistics are invariant under such transformations. Thereafter, the size of the two statistics (with respect to the yielded cutoff values) are calculated to affirm our results, after which we proceed to calculate the power of the tests with respect to Normal and Laplace distributed alternative data.

What follows is an exploration of the inversion theorem of the characteristic function to obtain probability density functions. As not all distributions admit a well-defined density (as will be exemplified in Section 3), it is important to verify the validity of this method; this is done in the setting of a Student's t-distributed random variable, as well as that of a sum of two such independent random variables. In the latter case, the convolution of the two independent densities is taken to be the parametric definition for the pdf. A non-parametric approach of pdf estimation, namely Gaussian kernel density estimation, is also explored for large samples of the aforementioned type. We further explore the complexities of this approach in accurately estimating densities of power tail distributions.

Thereafter, we explore the use of parametric and non-parametric bootstraps in the calculation of Expected shortfall (ES). ES is a risk measure that quantifies the loss one expects to observe beyond a certain confidence level usually denoted α, i.e. it is the expected loss given that it exceeds the α'th percentile of the loss distribution. The quantile that is said to be exceeded is referred to as the value at risk at the $1-\alpha$ level ($\text{VaR}_{(1-\alpha)}$). ES is a coherent risk measure. In particular, it satisfies the desirable property of sub-additivity, a feature not present in the VaR measure. The parametric and non-parametric bootstrap are powerful statistical tools in constructing confidence intervals for a statistic. Parametric bootstrapping is a resampling method that assumes the data follows a specified parametric distribution (e.g. $N(\mu, \sigma)$ or $t(\mu, \sigma, \nu)$). As such, maximum likelihood estimates (MLE) of the distribution parameters are calculated from the data. This MLE fitted distribution is then used to generate bootstrap re-samples, from which the desired statistic is calculated. After a sufficient number of bootstraps, the $\alpha/2$ and $1-\alpha/2$ quantiles of the calculated statistic yields the left and right bounds of the $(1-\alpha)*100\%$ confidence interval. The non-parametric bootstrap (also known as the empirical bootstrap) follows a similar methodology, but re-samples directly from the data instead of assuming an underlying distribution; the data sample is treated a representation of the true population of the data generating process.

2 Goodness of Fit Test with KD and AD

Suppose the body length of a species is normally distributed with unknown mean and variance. Now if we have a measurement of 100 individuals, and wish to determined whether this sample is randomly drawn from the population. This is the typical scenario where we can use goodness-of-fit test to make a statistical inference about the observed data, that is, to assess how "good" the data "fit" the assumed probability model.

There are multiple types of goodness-of-fit tests. While the Pearson's chi-square test, proposed by Pearson in 1900, is the oldest and most well known, it is not recommended for

the above-mentioned example problem, since when being used for continuous numerical data, it requires data to be artificially binned, but the test result is heavily dependent on the choice of such bins [SSSC68]. This study will instead focuses on the KD and its newer AD test. The idea of the tests is to compare ecdf, estimated from the data, with the theoretical cdf of the underlying continuous distribution to check if they well agree to each other.

The test statistics used by KD test is referred to as Kolmogorov–Smirnov distance. Let $y_1 < y_2 < \ldots < y_n$ be an ordered sequence of observations of an independent and identically distributed (i.i.d.) sample $\{X_1, X_2, \ldots, X_n\}$, then KD is defined as

$$KD = \max_i |\hat{F}_{emp}(y_i) - F(y_i)|, \tag{1}$$

where $\hat{F}_{emp}(y_i) = \frac{i-3/8}{n+1/4}$ is the ecdf of the observed data, and $F(y_i)$ is the null cdf evaluated at the data point. As will be further detailed below, we will be exploring a composite hypothesis, and will therefore take our null cdf to be $F(y_i) = \hat{F}_{fit}(y_i)$, i.e. the MLE fitted cdf based on our null hypothesis that the data is modelled by a particular family of distributions. The AD test statistic is similarly defined as

$$AD = \max_i \frac{|\hat{F}_{emp}(y_i) - F(y_i)|}{\sqrt{F(y_i)[1 - F(y_i)]}} \tag{2}$$

A showcase of implementing normality tests by simulation with KD and AD can be found in Paolella's book, specifically within Chapter 2 "Goodness of Fit and Hypothesis Testing" [Pao18]. This study aims to extend the established method for testing location-scale t-distribution:

H_0: the data consist of a i.i.d. sample drawn from some location-scale t-distribution.

H_a: It is not from a location-scale t-distribution.

The point of interest here is composite testing, therefore the null hypothesis doesn't specify the parameters of the location-scale t-distribution.

2.1 Simulation Methodology

Monte Carlo simulation has been employed throughout the study, with an objective to evaluate the powers of KD and AD statistics in testing whether a random sample of n independent observations come from a location-scale-t-distributed population. The levels of significance, α, were set as 0.01, 0.05 and 0.1.

The first step was to estimate the critical values for each of the significant levels for 3 different sample sizes ($n = 30, 60, 100$), and 3 different degree of freedom ($df = 3, 6, 12$), respectively. For each pair of (n, df), 100,000 independent student t samples ($\mu = 0, \sigma = 1$) were generated. We then employed maximum likelihood estimation (MLE) to estimate the parameters, i.e. df, μ, σ, for every sample, based on which the statistics KD and AD are obtained.

As KD and AD are right-tailed tests, the critical values are the $100(1-\alpha)th$ percentiles of the calculated test statistics. As a mean to validate the calculated critical values, we generated another 100,000 independent student t-distributed samples, and examined the percentage of their KD and AD values that were greater than the previously obtained critical values.

Finally, the powers of the KD and AD tests were evaluated against non-location-scale-t samples, also by simulation. The standard normal distribution and Laplace distribution with $\mu = 0, \lambda = 1$ were selected as representatives of non-location-scale-t-distributions, both of which are known to be lighter tailed than location-scale t-distribution. Again, 100,000 independent samples of size $n(30, 60, 100)$ were generated for each respectively. "Without" knowledge of these random samples, they would be assumed from location-scale t under the null hypothesis. Therefore we again used MLE to estimate their corresponding location-scale t parameters. Box constraints are recommended for the application of MLE in this situation [Pao18]. Multiple experimental trials with various constraints were conducted for glimpse into how the constraints would influence the efficiency and accuracy of the optimizer used in the MLE procedure, and

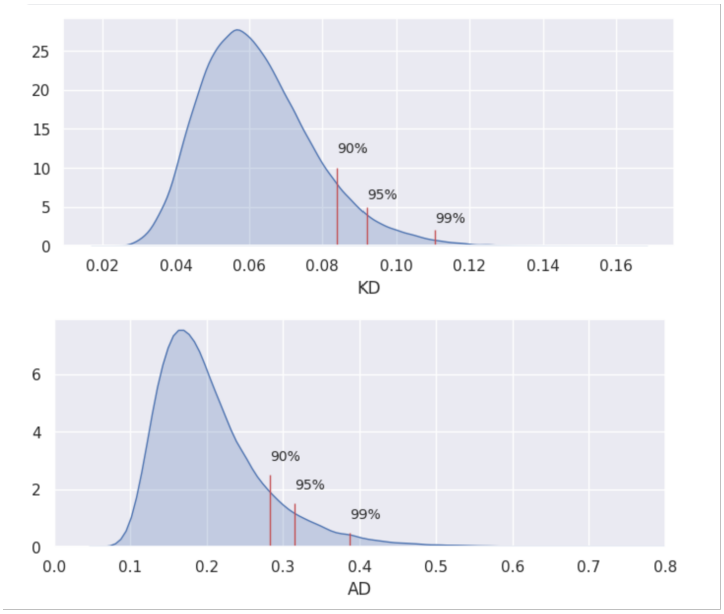


Figure 1: Estimations of the distributions of KD (upper) and AD (lower) for student t-distribution with $df = 6, n = 60$. Red vertical lines mark the 90th, 95th, 99th quantiles.

we ultimately arrived at

$$\begin{aligned} 0.00001 \leq df \leq 90, \\ -\infty < \mu < \infty, \\ 0.00001 < \sigma < \infty. \end{aligned}$$

Then the KD and AD test statistics following the same procedure. The percentage of the statistics exceeding the respective cutoff values gave the powers of (α-level) tests.

2.2 Results and Discussion

Both KD and AD, as random variables, are of positive skewness for location-scale t-distribution. This can be illustrated in histogram. Figure 1 shows their kernel density estimates based on the 100,000 independent samples that we used to estimate the critical values.

Table 1 lists the calculated critical values of the two tests for every combination of (df, n, α). It is clearly shown that the critical values at every α-level decreases as sample size n increases, indicating that the distributions of both statistics become narrower with larger sample size. This agrees with **Glivenko–Cantelli Theorem** that ecdf converges to true cdf in probability as $n \to \infty$ [Pao18]. What is of more interest is if the critical values vary with degree of freedom, as ideally we wish that the distributions of the KD and AS statistics are invariant in degree of freedom, otherwise it would be baseless to perform a composite test that encompasses the whole family of location-scale t-distributions.

From Table 1 we see that the KD critical values are fairly consistent across the three degrees of freedom, but not quite the case for AD, whose variation is noticeable. We recall that the AD statistics can be considered as a weighted form of the KD, and the weighting function is so constructed as to weigh more heavily on the tails of a distribution. We also note that the tails of the density functions of student t get much thinner when df increases from 3 to 12. These combined should account for the sensitivity of the AD we have observed in the study. To be more sure, We performed a paired student t tests to determine if the variation in AD is due to the degree of freedom or sampling randomness. It turned out that the difference is insignificant at 0.05 level.

The estimate results of the actual sizes of the KD and AD tests across all of the (df, n, α) combinations are presented in Table 2. We can see that all of the estimated values are suffi-

		KD			AD		
df	$n\backslash\alpha$	0.01	0.05	0.10	0.01	0.05	0.10
	30	0.1453	0.1212	0.1098	0.5000	0.4132	0.3774
3	60	0.1026	0.08746	0.08007	0.4577	0.3407	0.3003
	100	0.07977	0.06835	0.06320	0.4489	0.2922	0.2535
	30	0.1513	0.1264	0.1147	0.4623	0.3881	0.3507
6	60	0.1105	0.09279	0.08403	0.3868	0.3148	0.2826
	100	0.08532	0.07260	0.06651	0.3493	0.2691	0.2373
	30	0.1515	0.1280	0.1171	0.4979	0.4033	0.3574
12	60	0.1148	0.09616	0.08783	0.4574	0.3358	0.2919
	100	0.08974	0.07587	0.06973	0.4117	0.2890	0.2472

Table 1: Cutoff values for the KD and AD composite tests of location-scale t-distribution, as a function of sample size n, significance level α, and degree of freedom df, to four significant digits, based on simulation with 100 th replications.

ciently close to the respective α, which may assure the accuracy of our estimation of the cutoff values.

		KD			AD		
df	$n\backslash\alpha$	0.01	0.05	0.10	0.01	0.05	0.10
	30	0.0098	0.050	0.10	0.010	0.050	0.10
3	60	0.0098	0.048	0.099	0.0098	0.049	0.098
	100	0.011	0.050	0.099	0.010	0.049	0.099
	30	0.010	0.051	0.099	0.010	0.051	0.10
6	60	0.0098	0.050	0.10	0.010	0.049	0.098
	100	0.0094	0.049	0.10	0.0094	0.048	0.10
	30	0.0096	0.050	0.098	0.010	0.051	0.10
12	60	0.010	0.052	0.10	0.0096	0.051	0.099
	100	0.010	0.049	0.099	0.011	0.050	0.099

Table 2: Size of the KD and AD tests estimated via simulation, rounded to 2 significant digits.

Lastly but most importantly, we examine the results of estimated power of the KD and AD tests for location-scale t-distribution, as presented in Table 3 and 4. The first thing we note is that the powers of KD are considerably higher than those of AD in all cases. The difference is greater with Laplace distribution as the alternative than with Gaussian. we can confirm the following observations about the KD test:

- Power increases as sample size increases.
- Power increases at larger α-level test.
- No power is lower than α.

The larger the sample size, the more "accurate" estimation MLE is able to make for parameters against a sample, in the sense that it is to assign a large df value to the sample from underlying non-fat-tailed distributions, i.e. standard Gaussian and Laplace in our cases. This results in a higher probability for the test to identify a non-t-distribution, aka the power. Secondly, it is necessary that the estimated power is higher in a test of higher α-level , because by definition, the value of α is the probability of committing Type-1 error. When it increases, the probability of committing Type-2 error, $(1 - power)$, decreases, so power increases. Combining the third point, we might conjecture that the KD serve as an unbiased and consistent test statistics for the location-scale t-distribution.

The AD statistics, on the other hand, exhibits an odd behaviour that its power decreases as sample size increase. To confirm this, we carried out further experiment on sample size up

		KD			AD		
df	$n\backslash\alpha$	0.01	0.05	0.10	0.01	0.05	0.10
3	30	0.0259	0.106	0.184	0.00240	0.0187	0.0372
	60	0.0396	0.133	0.218	0.0002	0.00980	0.0281
	100	0.0548	0.152	0.235	0.0001	0.00680	0.0235
6	30	0.0186	0.0789	0.143	0.0070	0.0326	0.0698
	60	0.0230	0.0893	0.160	0.0020	0.0223	0.0528
	100	0.0264	0.0989	0.171	0.0006	0.0128	0.0352
12	30	0.0105	0.0527	0.105	0.0075	0.0334	0.0722
	60	0.0107	0.0555	0.110	0.0020	0.0210	0.0493
	100	0.0121	0.0587	0.116	0.0018	0.0115	0.0309

Table 3: Power of the KD and AD tests against standard normal distribution as the alternative, estimated via simulation, rounded to 3 significant digits.

		KD			AD		
df	$n\backslash\alpha$	0.01	0.05	0.10	0.01	0.05	0.10
3	30	0.0144	0.0576	0.112	0.0060	0.0469	0.0831
	60	0.0214	0.0778	0.146	0.0009	0.0257	0.0671
	100	0.0249	0.102	0.176	0.000	0.0119	0.0466
6	30	0.00910	0.0428	0.0866	0.0168	0.0696	0.133
	60	0.0112	0.0491	0.104	0.00890	0.0484	0.0998
	100	0.0136	0.0696	0.134	0.00130	0.0304	0.0781
12	30	0.0116	0.0492	0.0899	0.0145	0.0736	0.144
	60	0.0119	0.0562	0.1069	0.00410	0.0530	0.120
	100	0.0166	0.0749	0.135	0.00290	0.0320	0.0897

Table 4: Power of the KD and AD tests against standard Laplace distribution as the alternative, estimated via simulation, rounded to 3 significant digits.

to 1,000, and found that the power of the test drops consistently, except for very large degree of freedom, e.g., 90. In that case, the power of AD converges to one, suggesting that the AD test can work well with more normal-like distribution. We include this result in Table 7 and 8 in the Appendix.

Overall, the KD statistics outperforms the AD in terms of testing power, when applied to composite tests for the location-scale distribution. We have seen some (limited) evidence that the KD statistics is unbiased (power are greater than size of test) and consistent (power improves with an increase in sample size), but more experiments are required to confirm that. On the other side, the abnormal behaviour of the AD (power drops with an increase in sample size) could attract more research interest in the future. Lastly, this study only inspect the two statistics used for location-scale t distribution against symmetrical and non-fat-tailed alternative distribution, so how they would perform against skewed but fat-tailed distributions remains as an interesting topic for further investigation.

3 Obtaining Probability Density Functions

In many settings of research it is imperative to work with probability density functions, particularly when assuming that the data generating process follows a specific model with an associated pdf. Generally there are two approaches to calculating pdfs, namely by the assumption of an underlying model (parametric) or by a data driven method such as kernel density estimation (non-parametric). Models often have closed form expressions for their corresponding pdfs, making them relatively easy to compute. However, this is not guaranteed to be the case; the alpha stable distributions is an example of a family that does not have an analytic expression for its pdfs. Instead, the family is entirely characterised by its characteristic function. Nevertheless, we may still wish to utilize and calculate its pdf. Fortunately, this can be done by leveraging the inversion theorem, which states that the pdf, $f(x)$, of a continuous random variable can be obtained from its characteristic function via

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-ixt} \phi(t) dt, \tag{3}$$

where notably, the characteristic function is given by

$$\phi(t) = \mathbb{E}[e^{itX}] = \int_{-\infty}^{\infty} e^{itx} f(x) dx. \tag{4}$$

It is therefore clear that the pdf and cf are related via (inverse) Fourier transforms, and so they have a unique one-to-one correspondence (it is a well-known fact that the Fourier transform is bijective). Note that we have considered the case of a continuous random variable, but the theory is also valid in the discrete case.

3.1 The Student's t-distribution: Analytic PDF and the inverted CF

It is natural to posit the question of whether the inversion of the characteristic function yields accurate values for the pdf in practice. Thus, we explore the case of a continuous Student's t-distributed random variable $X \sim t_\nu$, which has a well defined pdf given by

$$f_X(x) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})} \left(1 + \frac{t^2}{\nu}\right)^{-(\nu+1)/2}, \tag{5}$$

where $\nu > 0$ and $x \in \mathbb{R}$. It's characteristic function, as derived in [Pao19], is also given by

$$\phi_X(t) = \frac{K_{\nu/2}(\nu^{1/2}|t|)(\nu^{1/2}|t|)^{\nu/2}}{\Gamma(\nu/2)2^{\nu/2-1}} \tag{6}$$

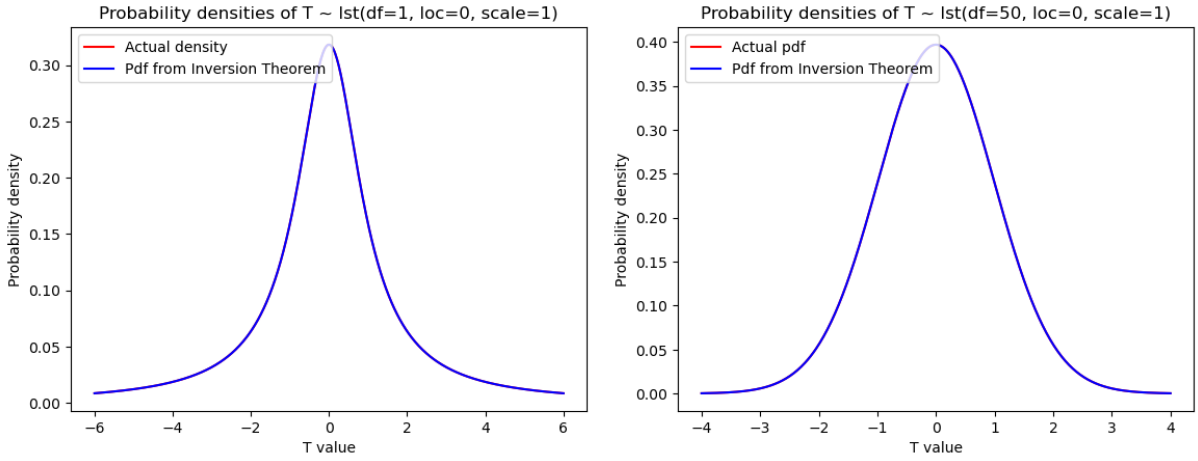
with special cases for $\nu = 3$ and $\nu = 5$, where

$$\phi_X(t;\nu=3) = (1+|t\sqrt{3}|)e^{-|t\sqrt{3}|} \quad \text{and}$$

$$\phi_X(t;\nu=5) = (1+|t\sqrt{5}|+\frac{5}{3}t^2)e^{-|t\sqrt{5}|}.$$

To verify the veracity of the inversion theorem approach we plot the well-defined pdf (5) as well as the pdf obtained by inversion in Figure 2. Note that the inversion is done via numerical integration of (3) with cf given by (6) utilizing the vector quadrature method as shown in the code listing below, where the characteristic function is also defined.

```
'''
Characteristic function of t-distribution
'''
def t_cf(df, t):
    # Avoid Bessel function
    if df == 3:
        return (1 + abs(t*np.sqrt(3))) * np.exp(-abs(t*np.sqrt(3)))

    elif df == 5:
        return (1 + abs(t*np.sqrt(5)) + 5/3 * t**2) * np.exp(-abs(t*np.sqrt(5)))

    # need to use Bessel function
    else:
        z = df/2
        x = np.sqrt(df) * abs(t)

        return (special.kv(z, x) * x**z) / (special.gamma(z) * 2**(z - 1))

'''
Numerical inversion of the characteristic function
'''
def inv_t_cf(df, x):
    # define kernel of inversion theorem
    f = lambda t, x, df: 1/(2*np.pi) * np.exp(-complex(0,1)*t*x) * t_cf(df, t) #

    # integrate numerically
    res = integrate.quad_vec(f, -np.inf, np.inf,  args=(x, df))
    res_val = res[0]

    return res_val
```



(a) (b)

Figure 2: Actual probability density function (red) and the probability density function obtained by the inversion theorem (blue) for degrees of freedom (a) $\nu = 1$ and (b) $\nu = 50$. Note that both pdfs are the same and essentially overlay one another. In the individual plot titles, df corresponds to ν, loc to μ and scale to σ. It is also worth noting that df and ν will be used throughout the text to refer to the degrees of freedom.

The plots appear to be identical for both the small and large degrees of freedom, indicating that they do indeed provide the same result no matter the ν value. The reader can verify that this is indeed the case by directly implementing the code in Appendix C.4. Now, with the confirmation of the inversion theorem's veracity, it is natural to extend our investigation to the sum of two independently distributed random variables.

3.2 PDF of the Sum of two independent Student's t-distributions

In theory, extending our setting to the case of a continuous random variable $Z = X+Y$ with X and Y being independently Student's t-distributed, i.e $X \sim t_{\nu_X}$ and $Y \sim t_{\nu_Y}$ with pdfs given by (5), should not affect the accuracy of the pdf generated by the inversion theorem. Here, the defining formula for the probability density of Z is

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(x) f_Y(z-x)dx = (f_X * f_Y)(z), \tag{7}$$

namely the convolution of the probability density of X with that of Y. Intuitively this makes sense, as the probability density of the sum being z will be the sum of the product of probability densities X and Y over all unique combinations of evaluated points of X and Y that sum up to z (over x, these points are X = x and Y = z -x for all possible $x \in \mathbb{R}$). The proof of this result is given in [JP04]. On the other hand, the inversion theorem and the corresponding Fourier transform relationships are now given by

$$f_Z(z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-izt} \phi_Z(t)dt \tag{8}$$

where

$$\phi_Z(t) = \mathbb{E}[e^{itZ}] = \int_{-\infty}^{\infty} e^{itz} f_Z(z)dz = \int_{-\infty}^{\infty} e^{itz}(f_X * f_Y)(z)dz = \phi_X(t)\phi_Y(t). \tag{9}$$

The last equality follows from the well-known result that the Fourier transform of a convolution is equal to the product of the individual Fourier transforms (the Convolution Theorem). Combining equation (7) and (8) then yields

$$f_Z(z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-izt} \phi_X(t)\phi_Y(t)dt. \tag{10}$$

The pdfs, as given by the convolution (6) and inversion theorem (9), can be computed numerically in Python by use of vector quadrature, as presented in the code listing below.

```
'''
Probability density of a sum of two independent t-distributed random variables via the convolution
'''
def sum_tconv_pdf(s, df1, df2, mu=0, sig=1):
    # via convolution
    f = lambda x, df1, df2, mu, sig, s: stats.t.pdf(x, df1, loc=mu, scale=sig) * stats.t.pdf(s - x, df2, loc=mu, scale=sig)

    res = integrate.quad_vec(f, -np.inf, np.inf, args=(df1, df2, mu, sig, s))
    y = res[0]

    return y

'''
Inversion of characteristic function of a sum of two independent t-distributed random variables
'''
def inv_tsum_cf(s, df1, df2):
    # kernel to integrate
    f = lambda t, s, df1, df2: 1 / (2*np.pi) * np.exp(-complex(0,1)*t*s) * t_cf(df1, t) * t_cf(df2, t)

    # inversion theorem (inverse fourier transform)
    res = integrate.quad_vec(f, -np.inf, np.inf, args=(s, df1, df2))
    res_val = res[0]
    return res_val
```

As of yet, only parametric approaches have been considered in calculating the pdf, whereby a particular distribution is assumed to model a random variable or data generating process. In our case, this was the Student's t-distribution. However, if the underlying model of a data set is not known or cannot be approximated accurately, a non parametric approach is feasible. An instinctive representation for the pdf is then the discrete histogram. It serves as a good initial estimate for the pdf, however, it is of course only defined in terms of discrete bins and does not represent a continuous probability density. A continuous extension of this method is the kernel density estimate (KDE), which is essentially a smoothed histogram, and is given by

$$\hat{f}_b(z) = \frac{1}{n}\sum_{i=1}^{n} K(\frac{z - z_i}{b}). \tag{11}$$

Here, the non-negative function K is called the kernel and the postive scalar b is known as the bandwidth, which controls the degree of smoothing. In our application it will be taken to be the standard normal distribution (this is the default choice of kernel in the scipy.stats library). Several other kernels are also commonly implemented, such as a uniform, triangular or Epanechnikov kernels (see [Epa69] for further details). The Gaussian kernel is a natural choice for smoothing due to its symmetric shape and decaying tails. For the choice of bandwidth, scipy implements the Scott's Rule by default and is what we utilise in our investigation. More details on this method and others can be seen in [Sco92].

For low sample sizes the kernel density estimate performs poorly, as expected. The question is whether the KDE obtains accurate results for large sample sizes. To answer this question, the KDE obtained from 100'000 samples of $Z = X + Y$ can be compared to the pdfs generated by both the convolution formula and the inversion of the cf, as is done in Figure 3 on the next page.

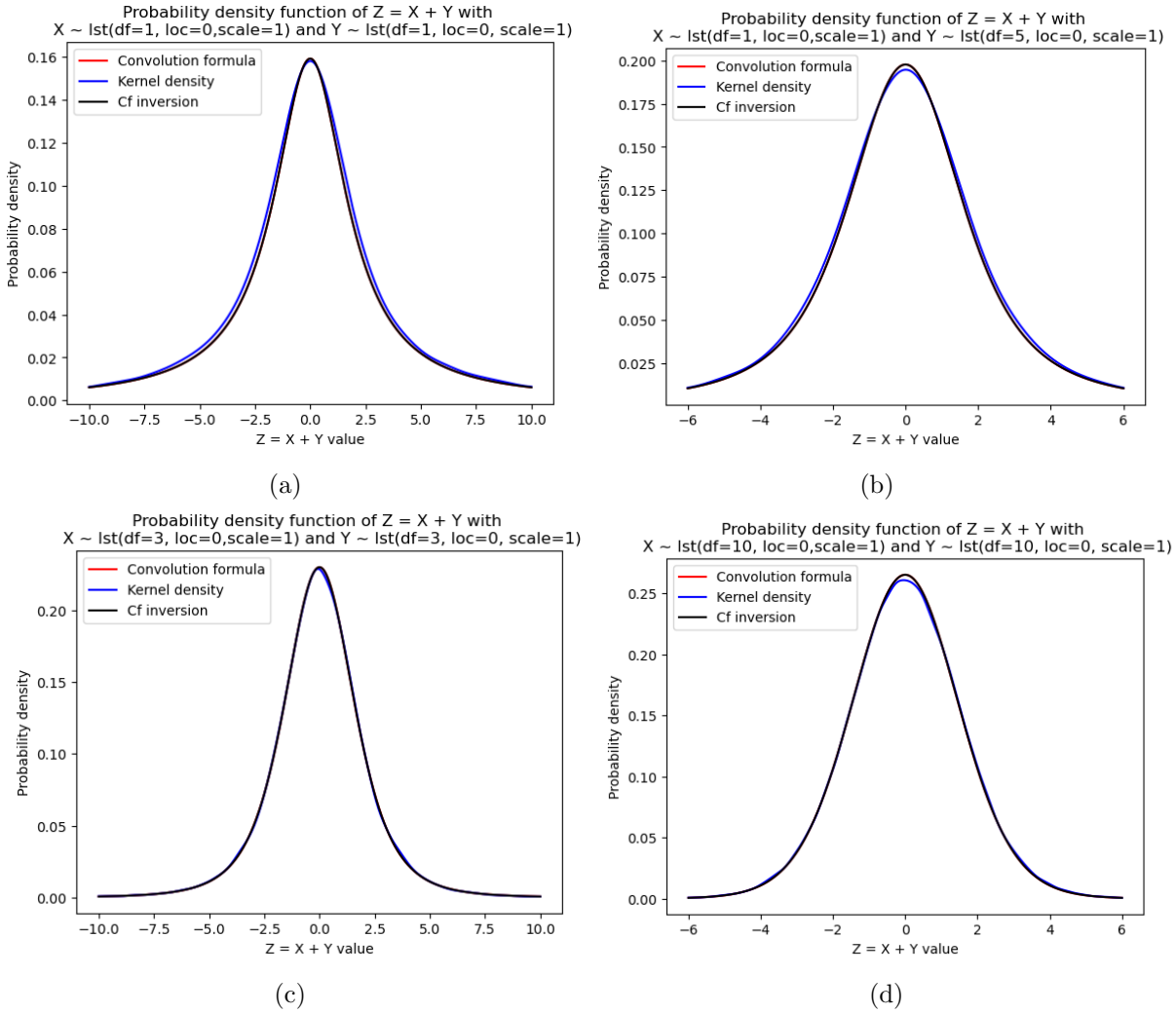


Figure 3: Probabilty density functions generated via the convolution formula (red), kernel density estimation (blue) and characteristic function inversion (black) for degrees of freedom of X and Y given by (a) df = 1, 1, (b) df = 1, 3, (c) df = 3, 3, and (d) df = 10, 10 respectively.

Results indicate that the convolution formula and cf inversion methods yield indistinguishable results, as expected given the theory and previous results in Figure 2. Indeed, this is observed across all ranges of degrees of freedom and their combinations (parameterising X and Y). The KDE is also seen to provide accurate fits in all cases, with a slightly improved fit for lower degrees of freedom, as the peak is not as effectively captured for the plots involving degrees of freedom greater than 5. Note, however, that the KDE procedure is modified in the case of data generated with low degrees of freedom (below 2.5). Instead of allowing all values to contribute to the KDE, tail values are not included and assigned a weight of zero; when the data is generated from a power tail distribution, the tail values will dominate the resulting density and significantly decrease the peak of the distribution. As a result, the data used in fitting the KDE is trimmed to exclude exorbitant tail values (see the full code in Appendix C.2). A cutoff of ± 25 is deemed appropriate to provide an accurate fit to the data, but more sophisticated methods can be implemented in determining an appropriate value (by further analysis of the data sample). The deviation in fit alternative cutoff values in the range of 25 were found to be relatively insignificant; choosing greater cutoff magnitudes resulted in a lower peak approximation and wider tails, whereas lower cutoff magnitudes resulted in a greater approximation for the peak and narrower tails. Despite the trimming of tail values in the case of $\nu_X, \nu_Y = 1$ and accurate fitting of the peak of the density, the KDE slightly overestimates the tail densities.

Hence, the accuracy of the KDE in the case of power tail distributions can perform well for large samples if an appropriate cutoff is chosen, and the convolution and inversion approaches

both plot the appropriate density. A further investigation of interest would be to concretely evaluate the effect of cutoff value and sample size in the accuracy of the KDE. An analysis of the performance for different kernels would also be beneficial.

4 Expected Shortfall

In the landscape of quantitative finance, where data-drive decision making is key, accurate assessment of risk measures holds great value. Among many commonly used metrics, risk management sector has growing attention on the ES, as it quantifies the potential loss beyond a certain threshold (Value at Risk). The ES is initially calculated analytically using the student t distribution. This provides an understanding of the ES with controlled and known conditions. Then we explore the bootstrap and compute confidence intervals to extend our methodology. These techniques align greatly with quantitative risk management as they provide tools to asses the uncertainty in the ES estimates.

4.1 Methodology

4.1.1 True Expected Shortfall

This part of the study focus on risk assessment and especially the expected shortfall (ES) for location-scale Student's t-distributed data, with data generating process $Z \sim lst(df, \mu, \sigma)$. The ES_t function (Appendix C.6) is created to compute the analytical expected shortfall for the data. It takes the inputs such as degrees of freedom (ν), mu (μ), sig (standard deviation, σ) and the tail probability for the ES calculation (ESLevel, which determines the VaR quantile referred to as q).

The algorithm includes input validations to ensure the validity of the parameter's. Specifically it checks that "ESLevel" is within the range of (0,1), that σ is greater than 0 and that v is larger than 1 to ensure finite first moments and the possibility of expected shortfall calculation.

The ES value is calculated using the analytical formula for a student t from [Pao18]. The critical value (c), which is also the value at risk, is determined by the percent-point function for the student t. The ES is then calculated using the probability density function, $\phi_v(c)$, and the cumulative distribution function, $\Phi_v(c)$. The function ES_t then outputs the computed true expected shortfall value. Refer to Appendix C.6-9 for all the relevant code.

$$\mathbb{E}[Z|Z < q] = -\frac{\phi_v(c)}{\Phi_v(c)} \cdot \left[\frac{v + \frac{c^2}{v}}{v - 1}\right] \tag{12}$$

4.1.2 The Bootstrap and CI

Next we want to develop this framework further by estimating the ES along with parametric and non-parametric confidence intervals. We here introduce the function "ES_conf_t (Appendix C.7), which takes as inputs the degrees of freedom (v), the simulation sample size (n), the number of bootstraps (B), the confidence level (α) and the tail probability ("ESLevel"). Our algorithm starts with input validation, making sure that the inputted degrees of freedom value is greater than one such that the expectation is finite. Additionally, the scale parameter is ensured to be greater than zero. This is followed by generating a random sample from the student's t-distribution as a simulated data set. The core methodology of this part involves the parametric and non-parametric bootstrapping:

Parametric Approach: The data is assumed to follow a location-scale Student's t distribution, and so the maximum likelihood estimates of the parameters are calculated, and the resulting MLE fitted distribution is taken to describe the data population. For each bootstrap iteration, new samples of data are generated from this distribution. From this new sample, the MLE estimates are calculated once more and used as input parameters to calculate the ES MLE by use of the analytical formula. This process is performed B times. After B iterations

the $1-\alpha$ confidence interval is created for the computed ES values by calculating the $\alpha/2$ and $1-\alpha/2$ quantiles for the B estimates, used as the left and right endpoints of the ci, respectively.

Non-parametric Approach: For each iteration a re-sampling with replacement is done from the originally simulated data sample. Essentially, this implies that the simulated data sample is take to represent the entire population of the data. The empirical ES estimate is then computed and the $1-\alpha$ confidence interval is similarly derived by using the ES estimates. The empirical ES is calculated using the formula from [Pao18], i.e. $\frac{1}{\xi}\overline{Y}_n \xrightarrow{p} \text{ES}(Z;\xi)$, where ξ is the ESLevel and $Y_i = Z_i \mathbb{1}(Z_i < q)$.

ES_conf_t then outputs the true ES the confidence intervals for both the parametric and non-parametric bootstrap.

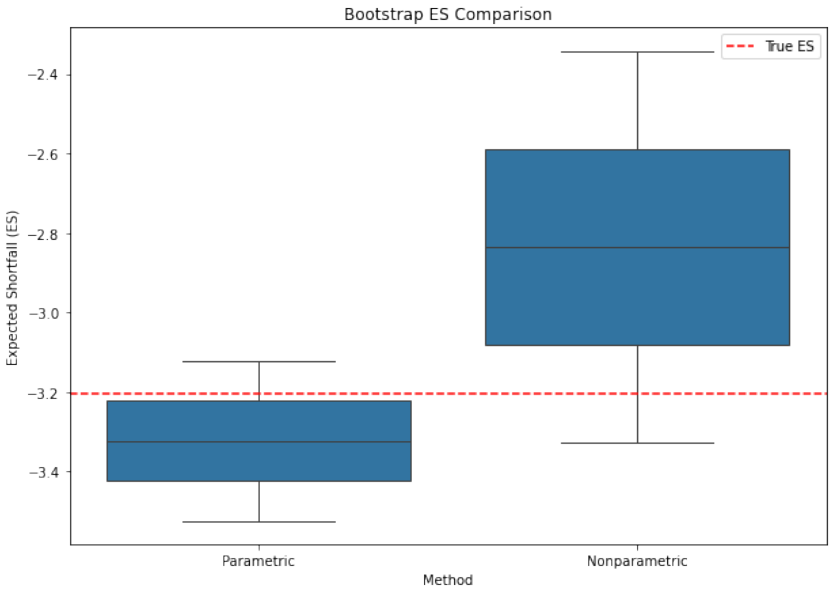


Figure 4: Comparison Parametric CI and Non-parametric CI

w

4.1.3 Coverage & CI length

To assess the actual coverage of the parametric and the non-parametric confidence intervals for the ES, we conduct sim simulations using the ES_conf_t function. Input parameters include degrees of freedom ν, number of simulations (sim), simulation sample size (n) and the number of bootstraps (B). For each iteration the true ES value, as well as the parametric and non-parametric confidence intervals are obtained. The average lengths of these are recorded, where the length of the CIs are calculated by subtracting the smallest value from the largest, e.g. len_par = ci_par[1] − ci_par[0] for the parametric case. The actual coverage is evaluated by assessing the proportion of times that the true ES value falls within the calculated confidence intervals.

4.2 Results and Discussion

In our study the true ES represents the analytical solution derived using the known parameters of the standard student t-distribution. This serves as our benchmark, reflecting the actual ES that is observed for our data generating process. We therefore expect the parametric bootstrapping will perform better than the non-parametric approach, given that the parametric assumption is true. Notably, the non-parametric approach can only leverage the empirical ES in its estimation, which only converges to the true solution for large enough samples. This presents an inherent flaw in the non-parametric approach; it uses the re-sampled data as a base for estimation of the ES, only capturing the variability in the sample data.

The table is calculated with with $df = 4$, $\alpha = 0.1$, $ESLevel = 0.05$ $n = 250$, and $B = 500$.

True ES	Parametric CI	Non-Parametric CI
−3.20287040	[−3.52354865, −3.12123748]	[−3.32790251, −2.34193127]

Table 5: Summary of ES and Confidence Intervals

Examining the results of the different estimation methods reveals that the parametric bootstrap, employing Maximum Likelihood Estimation (MLE) to assume a specific parametric distribution (the student t in our case) does indeed yield estimates closer to the true values in both mean and spread 4. This is an expected, given that the MLE tailors the assumed distribution to the observed data by maximizing the likelihood. This contributes to greater accuracy when the data aligns with the underlying parametric model. The non-parametric bootstrap, while advantageous in its flexibility and applicability to scenarios with unknown or complex distributions, is outperformed as a result of this fact. However, had our underlying parametric assumption been false, the non-parametric approach would likely have performed better. This is because both the MLE and the parametric bootstrap relies on a correctly specified underlying model and distribution. For a wrongly specified model the asymptotic properties of the MLE (asymptotically unbiased and efficient estimator) may not hold; Although the MLE is consistent and asymptotically efficient, samples must also be large enough to yield accurate estimates, and they are of course only valid if the assumed model is true.

In order to assess the performance and reliability of the estimated confidence intervals for the expected shortfall (ES) we repeat the process sim times (1000). For the parametric approach the actual coverage converges nicely to the confidence level $1 - \alpha$ with the predetermined values and also for somewhat higher degrees of freedom. The length of the confidence interval (CI) is quite narrow for the parametric compared to the non-parametric. This indicates higher precision in the estimates, that is, we have more confidence in the location of the true parameter value as the range of possible values gets narrower. This is to be expected based on the results from the box plot 4.

For the non-parametric approach the actual coverage is not as accurate which is inline with the results in the box plot. For $df = 4$ it underestimates the nominal coverage by a bit by being in the low 80's. For $df = 10$ the actual coverage is rather close and could be compared with the parametric numbers. The longer length for all of the non-parametric estimates indicates increased uncertainty and variability in the estimation of ES and we have lower precision in our estimates compared to the parametric approach.

The result from the table largely aligns with the theoretical expectations, which is that for a known distribution and low sample size the parametric approach shows it's advantages. It approaches the nominal coverage quite consistently, validating the previous distributional assumption that the data follows a student t distribution. Conversely, the non parametric approach, while slightly underestimating the coverage in some cases it remains a robust and adaptable tool for different scenarios with unknown distribution (which will be most of the cases outside simulations).

Approach	Actual Coverage	Length
Parametric, df=4,n=250	0.9	0.3987
Non-Parametric	0.822	1.4583
Parametric, df=4,n=500	0.875	0.2837
Non-Parametric	0.814	1.0569
Parametric,df=10,n=250	0.908	0.2605
Non-Parametric	0.885	0.8089
Parametric,df=30,n=250	0.897	0.2849
Non-Parametric	0.888	0.6505

Table 6: Actual Coverage and Length

For the simulation where we simulate over multiple sample sizes we get different results. The further exploration of the convergence for varying sample sizes is additionally explored

in in the appendix A.1 (6). The increase in performance for the non-parametric approach for larger sample sizes is also something one could expect as mentioned above the non-parametric bootstrap usually increases its performance with larger samples sizes.

To test our results further one could explore scenarios where the assumed parametric distribution diverges from the actual data distribution, assessing the robustness of the parametric approach. One could have done the same analysis for different distributions and/or done the test on real data where one cannot alter the data-generating process and the underlying distribution is unknown (in general). Lastly one could compare the expected shortfall with other common risk measures and compare their performance.

References

[Epa69] V.A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability and Its Applications*, 1969.

[JP04] J. Jacob and P. Protter. *Probability Essentials*. Springer-Verlag Berlin Heidelberg GmbH, 2004.

[Pao18] Marc S Paolella. *Fundamental Statistical Inference: A Computational Approach*. John Wiley & Sons, 2018.

[Pao19] Marc S Paolella. *Linear Models and Time-Series Analysis Regression, ANOVA, ARMA, and GARCH*. John Wiley & Sons, 2019.

[Sco92] D.W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. ohn Wiley Sons, Inc., 1992.

[SSSC68] M. B. Wilk S. S. Shapiro and H. J. Chen. A comparative study of various tests for normality. *Journal of the American Statistical Association*, 63(324):1343–1372, 1968.

A Plots

A.1 Convergence of Actual Coverage in Sample Size

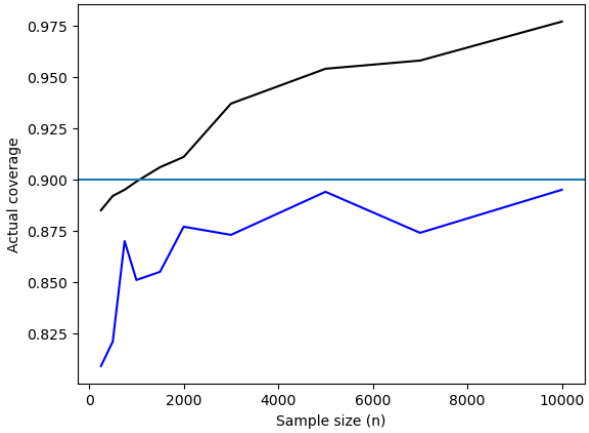


Figure 5: df = 4, sim = 1000, B = 500.

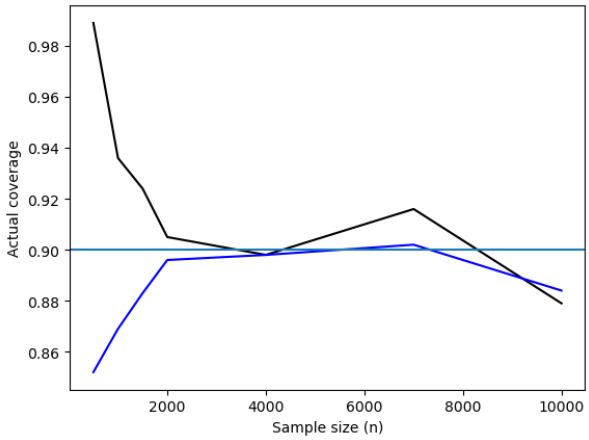


Figure 6: df = 10, sim = 1000, B = 500.

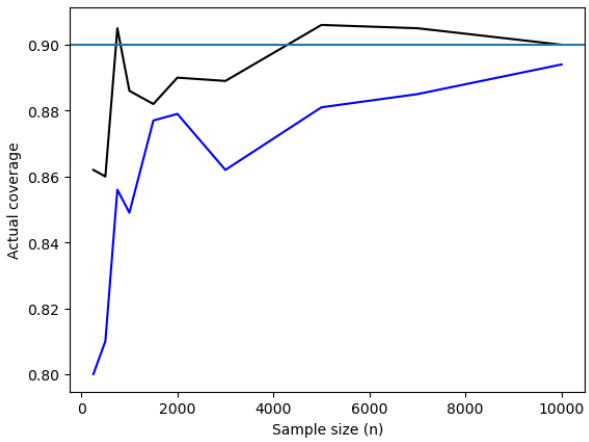


Figure 7: df = 12, sim = 1000, B = 500.

B Tables

		KD			AD		
df	$n\backslash\alpha$	0.01	0.05	0.10	0.01	0.05	0.10
3	30	0.0259	0.106	0.184	0.00240	0.0187	0.0372
	60	0.0396	0.133	0.218	0.0002	0.00980	0.0281
	100	0.0548	0.152	0.235	0.0001	0.00680	0.0235
	1000	0.0746	0.196	0.290	0.0003	0.0065	0.0247
6	30	0.0186	0.0789	0.143	0.0070	0.0326	0.0698
	60	0.0230	0.0893	0.160	0.0020	0.0223	0.0528
	100	0.0264	0.0989	0.171	0.0006	0.0128	0.0352
	1000	0.0515	0.152	0.242	0.0003	0.0111	0.0298
12	30	0.0105	0.0527	0.105	0.0075	0.0334	0.0722
	60	0.0107	0.0555	0.110	0.005	0.0400	0.0823
	100	0.0121	0.0587	0.116	0.0018	0.0115	0.0309
	1000	0.0428	0.166	0.270	0.0001	0.0014	0.0042
90	30	0.0102	0.0500	0.103	0.0068	0.0436	0.094
	60	0.0099	0.0506	0.101	0.0020	0.0210	0.0493
	100	0.0082	0.0480	0.0982	0.0065	0.0375	0.0791
	1000	0.0089	0.0489	0.0994	0.0028	0.0227	0.0520

Table 7: Power of the KD and AD tests against standard normal distribution as the alternative, estimated via simulation, rounded to 3 significant digits.

		KD			AD		
df	$n\backslash\alpha$	0.01	0.05	0.10	0.01	0.05	0.10
3	30	0.0144	0.0576	0.112	0.0060	0.0469	0.0831
	60	0.0214	0.0778	0.146	0.0009	0.0257	0.0671
	100	0.0249	0.102	0.176	0.000	0.0119	0.0466
	1000	0.500	0.761	0.860	0.000	0.0009	0.0353
6	30	0.00910	0.0428	0.0866	0.0168	0.0696	0.133
	60	0.0112	0.0491	0.104	0.00890	0.0484	0.0998
	100	0.0136	0.0696	0.134	0.00130	0.0304	0.0781
	1000	0.386	0.679	0.806	0.000	0.0035	0.0493
12	30	0.0116	0.0492	0.0899	0.0145	0.0736	0.144
	60	0.0119	0.0562	0.1069	0.00410	0.0530	0.120
	100	0.0166	0.0749	0.135	0.00290	0.0320	0.0897
	1000	0.203	0.527	0.691	0.000	0.001	0.0263
90	30	0.104	0.245	0.352	0.162	0.3527	0.4762
	60	0.229	0.442	0.568	0.2647	0.5031	0.6428
	100	0.384	0.642	0.758	0.3576	0.6244	0.7609
	1000	1.000	1.000	1.000	0.928	1.000	1.000

Table 8: Power of the KD and AD tests against standard Laplace distribution as the alternative, estimated via simulation, rounded to 3 significant digits.

C Python Code

C.1 KD and AD Statistics

```
import numpy as np
import pandas as pd
from scipy.stats import t, norm, laplace
from scipy.optimize import minimize
from scipy.special import beta
import seaborn as sns
import matplotlib.pyplot as plt

def t_KS_AS(x, t_param):
    '''
    This function calculates the Kolmogorov{Smirnov and Anderson{Darling
    statistics for given a location-scale t sample x with parameters t_param.
    '''
    x = np.sort(x)
    n = len(x)
    df, mu, scale = t_param
    cdf_fit = t.cdf(x, df=df, loc=mu, scale=scale)  # Fitted CDF
    ecdf = (np.arange(1, n+1) - 3/8) / (n + 1/4)    # Empirical CDF
    KS = max(abs(cdf_fit - ecdf))    # Kolmogorov{Smirnov
    AS = np.max(abs(ecdf - cdf_fit) / ...
        np.sqrt(cdf_fit * (1 - cdf_fit)))  # Anderson{Darling
    return [KS, AS]

def t_NegLogLik(param, x):
    '''
    This function computes the negative log likelihood function
    for location-scale t distribution.
    '''
    df, mu, scale = param
    K = beta(df/2, 0.5) * np.sqrt(df)
    z = (x - mu) / scale
    loglik = -np.log(scale) - np.log(K) - ...
        ((df+1)/2)*np.log(1+np.multiply(z,z)/df)
    negloglik = -np.sum(loglik)
    return negloglik
```

C.2 Estimation of Cutoff Values

```
# Estimate of cutoff values for every combination of
# (alpha, df, n)
np.random.seed(8001)
alphas = [.01, .05, .1]
sample_sizes = [30, 60, 100]
dofs = [3, 6, 12]
num_samples = int(1e+5)

df_KS = pd.DataFrame(index=alphas)
df_AS = pd.DataFrame(index=alphas)
KS = np.zeros([len(dofs), len(sample_sizes), num_samples])
AS = np.zeros([len(dofs), len(sample_sizes), num_samples])
KS_cutoffs = np.zeros([len(dofs), len(sample_sizes), len(alphas)])
AS_cutoffs = np.zeros([len(dofs), len(sample_sizes), len(alphas)])

for dof, d in zip(dofs, range(len(dofs))):
    for sample_size, n in zip(sample_sizes, range(len(sample_sizes))):
        sim = t.rvs(df = dof, size = num_samples*sample_size)
```

```
        sim = sim.reshape(sample_size, num_samples)
        sim = np.sort(sim, axis=0)
        bnds = ((1, None), (None, None), (0.01, 50))
        init_guess = np.array([dof, 0, 1])
        col_name = str((dof, sample_size))
        for i in range(num_samples):
            mle = minimize(t_NegLogLik, x0=init_guess, args=sim[:,i],
                           tol=1e-5, bounds=bnds, method='L-BFGS-B')
            [KS[d,n,i], AS[d,n,i]] = t_KS_AS(sim[:,i], mle.x)
        KS_cutoffs[d,n,:] = np.quantile(KS[d,n,:], 1-np.array(alphas))
        AS_cutoffs[d,n,:] = np.quantile(AS[d,n,:], 1-np.array(alphas))
        df_KS[col_name] = KS_cutoffs[d,n,:]
        df_AS[col_name] = AS_cutoffs[d,n,:]

np.save("KS_cutoffs.npy", KS_cutoffs)
np.save("AS_cutoffs.npy", AS_cutoffs)
df_KS.to_csv("KS_cutoffs.csv", index=True)
df_AS.to_csv("AS_cutoffs.csv", index=True)


# Confirm the sizes of the KD, AD test
KS_cutoffs = np.load('KS_cutoffs.npy')
AS_cutoffs = np.load('AS_cutoffs.npy')
np.random.seed(8002)
alphas = [.01, .05, .1]
sample_sizes = [30, 60, 100]
dofs = [3, 6, 12]
num_samples = int(1e+5)
KS_TestSize = np.zeros([len(dofs), len(sample_sizes), len(alphas)])
AS_TestSize = np.zeros([len(dofs), len(sample_sizes), len(alphas)])
df_KS_testsize = pd.DataFrame(index=alphas)
df_AS_testsize = pd.DataFrame(index=alphas)

for dof, d in zip(dofs, range(len(dofs))):
    for sample_size, n in zip(sample_sizes, range(len(sample_sizes))):
        KS = np.zeros(num_samples)
        AS = np.zeros(num_samples)
        sim = t.rvs(df = dof, size = num_samples*sample_size)
        sim = sim.reshape(sample_size, num_samples)
        sim = np.sort(sim, axis=0)
        bnds = ((1, 50), (None, None), (0.01, 50))
        init_guess = np.array([dof, 0, 1])
        col_name = str((dof, sample_size))

        for i in range(num_samples):
            mle = minimize(t_NegLogLik, x0=init_guess, args=sim[:,i],
                           tol=1e-5, bounds=bnds, method='L-BFGS-B')
            [KS[i], AS[i]] = t_KS_AS(sim[:,i], mle.x)

        for i in range(len(alphas)):
            KS_TestSize[d,n,i] = sum(KS > KS_cutoffs[d,n,i])/KS.shape[0]
            AS_TestSize[d,n,i] = sum(AS > AS_cutoffs[d,n,i])/AS.shape[0]

        df_KS_testsize[col_name] = KS_TestSize[d,n,:]
        df_AS_testsize[col_name] = AS_TestSize[d,n,:]

df_KS_testsize.to_csv("KS_testsize.csv", index=True)
df_AS_testsize.to_csv("AS_testsize.csv", index=True)
```

C.3 Evaluation of Power

```
# Evaluate powers of the tests using Gaussian and Laplace as example alternatives
np.random.seed(8003)
alphas = [.01, .05, .1]
sample_sizes = [30, 60, 100]
dofs = [3, 6, 12]
num_samples = int(1e+5)
Norm_KSpw = np.zeros([len(dofs), len(sample_sizes), len(alphas)])
Lap_KSpw = np.zeros([len(dofs), len(sample_sizes), len(alphas)])
Norm_ASpw = np.zeros([len(dofs), len(sample_sizes), len(alphas)])
Lap_ASpw = np.zeros([len(dofs), len(sample_sizes), len(alphas)])
df_KS_normpw = pd.DataFrame(index=alphas)
df_AS_normpw = pd.DataFrame(index=alphas)
df_KS_lappw = pd.DataFrame(index=alphas)
df_AS_lappw = pd.DataFrame(index=alphas)

for dof, d in zip(dofs, range(len(dofs))):
    for sample_size, n in zip(sample_sizes, range(len(sample_sizes))):

        KS_norm = np.zeros(num_samples)
        AS_norm = np.zeros(num_samples)
        KS_lap = np.zeros(num_samples)
        AS_lap = np.zeros(num_samples)

        sim_norm = norm.rvs(size = num_samples * sample_size)
        sim_norm = sim_norm.reshape(sample_size, num_samples)
        sim_norm = np.sort(sim_norm, axis=0)
        sim_lap = laplace.rvs(size = num_samples * sample_size)
        sim_lap = sim_lap.reshape(sample_size, num_samples)
        sim_lap = np.sort(sim_lap, axis=0)
        col_name = str((dof, sample_size))

        for i in range(num_samples):
            bnds = ((1, 90), (None, None), (0.01, 50))
            init_guess = np.array([dof, 0, 1])
            mle_norm = minimize(t_NegLogLik, x0=init_guess, args=sim_norm[:,i],
                                tol=1e-5, bounds=bnds, method='L-BFGS-B')
            [KS_norm[i], AS_norm[i]] = t_KS_AS(sim_norm[:,i], mle_norm.x)
            mle_lap = minimize(t_NegLogLik, x0=init_guess, args=sim_lap[:,i],
                               tol=1e-5, bounds=bnds, method='L-BFGS-B')
            [KS_lap[i], AS_lap[i]] = t_KS_AS(sim_lap[:,i], mle_lap.x)

        for i in range(len(alphas)):
            Norm_KSpw[d,n,i] = sum(KS_norm > KS_cutoffs[d,n,i]) / KS_norm.shape[0]
            Norm_ASpw[d,n,i] = sum(AS_norm > AS_cutoffs[d,n,i]) / AS_norm.shape[0]
            Lap_KSpw[d,n,i] = sum(KS_lap > KS_cutoffs[d,n,i]) / KS_lap.shape[0]
            Lap_ASpw[d,n,i] = sum(AS_lap > AS_cutoffs[d,n,i]) / AS_lap.shape[0]

        df_KS_normpw[col_name] = Norm_KSpw[d,n,:]
        df_AS_normpw[col_name] = Norm_ASpw[d,n,:]
        df_KS_lappw[col_name] = Lap_KSpw[d,n,:]
        df_AS_lappw[col_name] = Lap_ASpw[d,n,:]

df_KS_normpw.to_csv("KS_normpw.csv", index=True)
df_AS_normpw.to_csv("AS_normpw.csv", index=True)
df_KS_lappw.to_csv("KS_lappw.csv", index=True)
df_AS_lappw.to_csv("AS_lappw.csv", index=True)
```

```
57
```

C.4 PDF plots of Student's t-distribution

Below is the code implemented in Section 3.1 for generating the closed form pdf and inverted cf plots.

```
'''
Plots of the  actual density of the t-distribution and the one generated via
↪  inversion of the characteristic function
'''
def t_pdf(df, mu = 0, sig = 1):
    # check df is positive
    if df <= 0:
        print('ERROR: The degrees of freedom parameter must be greater than
        ↪  zero')
        return

    # // plot between -4 to 4 for df > some value. Else plot between -6 to 6
    plt.figure()
    plt.title(f'Probability densities of T ~ lst(df={df}, loc={mu},
    ↪  scale={sig})')

    if df > 5:
        x = np.linspace(-4, 4, 500)

        # Actual
        y_act = stats.t.pdf(x, df=df, loc=mu, scale=sig)
        plt.plot(x, y_act, label='Actual pdf', color='r')

        # Via inversion
        y_inv = inv_t_cf(df, x)
        plt.plot(x, y_inv, label='Pdf from Inversion Theorem', color='b')


    else:
        x = np.linspace(-6, 6, 500)

        # Actual
        y_act = stats.t.pdf(x, df=df,
                            loc=mu, scale=sig)
        plt.plot(x, y_act, label='Actual density', color='r')

        # Via inversion
        y_inv = inv_t_cf(df, x)
        plt.plot(x, y_inv, label='Pdf from Inversion Theorem', color='b')

    plt.legend(loc='upper left')
    plt.ylabel('Probability density')
    plt.xlabel('T value')
    plt.show()

    return
```

C.5 PDF plots for the Sum of two independently distributed Student's t-distributed random variables

The code implemented in calculating the PDF via convolution, inversion of the characteristic function, and kernel density estimation. Note that the code listing in Section 3.2 is also required

to define the functions sum_tconv_pdf and inv_tsum_cf (for calculating the convolution and inversion).

```
def t_sum_pdf(df1, df2, mu = 0, sig = 1):
    if (df1 < 5) and (df2 < 5):
        s = np.linspace(-10, 10, 1000)
    else:
        s = np.linspace(-6, 6, 1000)

    plt.figure()
    plt.title(f'Probability density function of Z = X + Y with \n X ~
    ↪  lst(df={df1}, loc=0,scale=1) and Y ~ lst(df={df2}, loc=0, scale=1)')

    # via convolution
    z_conv = sum_tconv_pdf(s, df1, df2, mu, sig)

    plt.plot(s, z_conv, color = 'r', label='Convolution formula')

    # simulation and kernel density
    x = stats.t.rvs(df=df1, loc=mu, scale=sig, size= 100000)
    y = stats.t.rvs(df=df2, loc=mu, scale=sig, size= 100000)

    z_sim = x + y

    # given the large tails of low df t-distributions, only consider sample
    ↪  values in a given range
    # when constructing the kernel density estimate
    if (df1 <= 2.5) or (df2 <= 2.5):
        z_sim = np.sort(z_sim)
        indices = (-25 <= z_sim) & (z_sim <= 25)
        w = indices / len(indices)
        density = stats.gaussian_kde(z_sim, weights=w)
    else:
        density = stats.gaussian_kde(z_sim)

    plt.plot(s, density(s), color='b', label='Kernel density')


    # characteristic inversion
    y_inv = inv_tsum_cf(s, df1, df2)
    plt.plot(s, y_inv, color='k', label='Cf inversion')

    plt.legend(loc='upper left')
    plt.xlabel('Z = X + Y value')
    plt.ylabel('Probability density')
    plt.show()


    return
```

C.6 True expected shortfall

```
def ES_t(df, mu = 0, sig = 1, ESLevel = 0.05):
    # Check input values
    if (ESLevel < 0) or (ESLevel > 1):
        print('ERROR: ESLevel must be in (0, 1)')
        return
    if (sig <= 0):
        print('ERROR: Scale parameter must be > 0')
        return
```

```
    if (df <= 1):
        print('ERROR: Degrees of freedom must be greater than 1 for finite first moment')

    # Computing ES via analytic formula
    c = stats.t.ppf(ESLevel, df = df, loc = mu, scale = sig)
    ES = -(stats.t.pdf(c, df=df, loc=mu, scale=sig) /
    stats.t.cdf(c, df=df, loc=mu, scale=sig)) * (df + c**2) / (df - 1)

    return ES
```

C.7 ES bootstrap CI MLE

```
def ES_conf_t(df, n = 500, B = 500, alpha = 0.1, ESLevel = 0.05):
    # Input checks
    if df <= 1:
        print('ERROR: Degrees of freedom must be greater than 1 for finite first moment')
        return
    if (ESLevel < 0) or (ESLevel > 1):
        print('ERROR: ESLevel must be in (0, 1)')
        return

    # True ES value
    ES = ES_t(df = df, ESLevel = ESLevel)

    # Sample data
    samp = stats.t.rvs(df = df, size = n) # loc, scale = 0, 1


    ## Bootstrapping
    ES_est_par = np.zeros(B)
    ES_est_npar = np.zeros(B)

    # Parametric
    # no closed formed expression for MLEs. Numerically optimize:
    bounds = [(-np.inf, np.inf), (1e-5, np.inf), (1e-5, np.inf)]
    params0 = [0, 1, df]
    res = optimize.minimize(lst_loglikelihood, params0, method='L-BFGS-B', args=samp, bounds=bounds, tol=1e-3) # 1e-6

    # MLEs
    mu_mle, sig_mle, df_mle = res.x

    for b in range(B):
        # resample from mle fitted distribution
        resamp = stats.t.rvs(df = df_mle, loc = mu_mle, scale = sig_mle, size = n)

        # MLE fit again
        res2 = optimize.minimize(lst_loglikelihood, params0, method='L-BFGS-B', args=resamp, bounds=bounds, tol=1e-3) #1e-6

        # MLEs
        mu_mle_sim, sig_mle_sim, df_mle_sim = res2.x

        # Analytic ES value
        ES_est = ES_t(df = df_mle_sim, mu = mu_mle_sim, sig = sig_mle_sim, ESLevel = ESLevel)
        # Store
```

```
        ES_est_par[b] = ES_est

    # Get 90% parametric ci
    lb_par = np.quantile(ES_est_par, alpha/2)
    ub_par = np.quantile(ES_est_par, 1 - alpha/2)

    ci_par = [lb_par, ub_par]

    # Non parametric
    for b in range(B):
        # resample with replacement
        resamp = np.random.choice(samp, size = n)

        # Calculate empirical ES estimate
        # VaR cutoff
        VaR = np.quantile(resamp, ESLevel)

        # tail val indicator
        I_tailval = 1*(resamp < VaR)

        # empirical estimate
        ES_emp = (1/ESLevel) * np.mean(resamp * I_tailval)
        ES_est_npar[b] = ES_emp


    # Get 90% non parametric ci
    lb_npar = np.quantile(ES_est_npar, alpha/2)
    ub_npar = np.quantile(ES_est_npar, 1 - alpha/2)

    ci_npar = [lb_npar, ub_npar]

    return (ES, ci_par, ci_npar)
```

C.8 Coverage and CI Length

```
def coverages(df, sim=1000, n=500, B=500):
    par_cover = np.zeros(sim)
    par_lengths = np.zeros(sim)
    npar_cover = np.zeros(sim)
    npar_lengths = np.zeros(sim)

    for i in range(sim):
        # simulate
        res = ES_conf_t(df=df, n = n, B = B) # should the parametric mle
        ↪  estimates be kept the same for each simulation?

        # true ES, parametric ci, and non parametric ci
        ES_true = res[0]
        ci_par = res[1]
        ci_npar = res[2]

        # lengths of the confidence intervals
        len_par = ci_par[1] - ci_par[0]
        par_lengths[i] = len_par
        len_npar = ci_npar[1] - ci_npar[0]
        npar_lengths[i] = len_npar

        if (ES_true <= ci_par[1]) and (ES_true >= ci_par[0]):
            par_cover[i] = 1

```

```
        if (ES_true <= ci_npar[1]) and (ES_true >= ci_npar[0]):
            npar_cover[i] = 1


    # actual coverages
    par_acov = np.mean(par_cover)
    print(f'Parametric 90% nominal, actual convergence: {par_acov}')
    par_avg_len = np.mean(par_lengths)
    print(f'average length: {par_avg_len} \n')

    npar_acov = np.mean(npar_cover)
    print(f'Non parametric 90% nominal, actual convergence: {npar_acov}')
    npar_avg_len = np.mean(npar_lengths)
    print(f'average length: {npar_avg_len} \n')

    return [[par_acov, par_avg_len], [npar_acov, npar_avg_len]]

```

C.9 Coverage for multiple sample sizes

```
ns = [500, 1000, 1500, 2000, 4000, 7000, 10000]

p_acovs = np.array([])
np_acovs = np.array([])

for n in ns:
    print(n)
    sim_results = coverages(df = 10, sim = 1000, n = n) #try more sim /different
    ↪  df
    p_acovs = np.append(p_acovs, sim_results[0][0])
    np_acovs = np.append(np_acovs, sim_results[1][0])

```