Faculty of Business, Economics and Informatics

Statistical Foundations for Finance

Prof. Dr. Marc Paolella

Assignment 2

Authors:	Broennimann, Nicoletta	16-738-148
	Jezler, Linda	15-703-333
	Villiger, Cesare	17-920-208

In Partial Fulfillment of the Requirements for the Course "Statistical Foundations for Finance (Mathematical and Computational Statistics with a View Towards Finance)".

Date of Submission
November 13, 2022

Contents

1 Parametric and Non-Parametric Bootstrap of ES based on Student t Distribution (Student t Assumption in Param. Bootstrap)

In question 1 we define three true parameters (degrees of freedom, location and scale) and simulate a T-length sequence of IID Student t random variates. Based on these simulations, we use the bootstrap technique to find a 90% confidence interval (CI) for the empirical expected shortfall (ES) based on B bootstrap replications. We then report the actual coverage probability and the average lengths of the confidence intervals as a function of T for both the parametric and non-parametric bootstrap.

1.1 Code

Depending on whether the parametric or non-parametric bootstrap is applied, we comment out the respective lines (line 61 is non-parametric and line 62 is parametric). The same goes for the method of maximum likelihood estimation (MLE): lines 50-52 are for the MLE using the built-in Matlab function, lines 56-57 for the MLE using the function provided in the statistical inference book, which is supplied in the Appendix (5.3). We redefine the parameter estimates that are yielded by each function, because they do not return the parameters in the same order. Of course, we could have changed the according functions, but we found it easier this way; the reader needs to pay attention to which parameter we are discussing. We use the simpler implementation of the MLE function from the book, because we did not observe much difference in the outputs of the programs and it turned out to be a bit slower. Given our limited computing resources, we thought it best to take the trade-off of bigger sample sizes and repetitions at the cost of slightly less accurate MLEs. If the reader wishes to change the program below to use the more sophisticated MLE function, simply change 'tmaxlik0' to 'tmaxlik' and accommodate for the fact that the output of the latter consists also of the standard errors of the estimated parameters. For completeness, the more sophisticated function is provided in the Appendix (5.5).
Additionally, we loop over all values of T and append the values in a nice vector that facilitates the copying to Latex. All codes are structured so that looping is possible, but if one wishes to run a simple specification, all variables are defined and commented out in the inner part of the loop, such that it could be copied to a new script and run individually.

```
% =========================Exercise 1================================
% Simulate T-length vectors of IID location scale student t data
% and use parametric and non-parametric bootstrap to check whether
% the true ES in the CI. This yields the actual coverage and the
% average lengths of the CI.
% ==================================================================

% The output for each T will be appended to these vectors
acc = [];
CI = [];

% Loop over all values of T
for T = [500 1000 2000]

    % Define Variables
    rand('twister',6) % set seed value so we can replicate results
    rep = 300;
    p = 0.9; alpha_CI = 0.1; alpha_VaR = 0.01;
    loc = 1; scale = 2; df = 4; % set parameters for student t
    % distribution
    initvec = [df loc scale]; % needed for tlikmax0
    B1 = 1e3; ESvec = zeros(B1,1); %set parameters for bootstrap
    bool_acc = zeros(rep,1); length_ci = zeros(rep,1);% vector for
    % coverage accuracy and CI length

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Compute True ES to check if it is in CI
    c01 = tinv(alpha_VaR , df); % left tail quantile, for loc-0
    % scale-1
    truec = loc+scale*c01; % left tail quantile c
    ES01 = -tpdf(c01,df)/tcdf(c01,df) * (df+c01^2)/(df-1);
    trueES = loc+scale*ES01;


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    % Loop over repetitions
    for i=1:rep

      % Create random variates of student t distribution
      % and calculate empirical
      % ES for comparison with true ES with bootstrap
      data = loc+scale*trnd(df,T,1); % random variate for
      % student t distribution

      % Calculation of MLE for parametric bootstrap with matlab
      %function, output: loc scale df
      %theta_mle_matlab = mle(data,'Distribution','tLocationScale');
      %loc_mle = theta_mle_matlab(1); scale_mle
      %= theta_mle_matlab(2); df_mle = theta_mle_matlab(3);

      % Calculation of MLE for parametric bootstrap with
      % function from the book, output: df loc scale
      theta_mle_book = tlikmax0(data, initvec);
      loc_mle = theta_mle_book(2); scale_mle = ...
```

```
            theta_mle_book(3); df_mle = theta_mle_book(1);

        % bootstrap
        for b1=1:B1
            ind = unidrnd(T,[T,1]); bootsamp=data(ind); % nonpara boot
            %bootsamp = loc_mle+scale_mle*trnd(df_mle,T,1); % para boot
            VaR = quantile(bootsamp, alpha_VaR);
            temp = bootsamp(bootsamp≤VaR); ESvec(b1)=mean(temp);
        end
        ci = quantile(ESvec,[alpha_CI/2 1-alpha_CI/2]);
        bool_acc(i) = (trueES>ci(1)) & (trueES<ci(2)); % check
        %whether the true value is in the CI
        length_ci(i) = ci(2)-ci(1); % calculate the length of the CI

    end
    length90 = mean(length_ci); % calculate mean length of CI
    actual90 = mean(bool_acc); % calculate actual coverage
    %probability

    % Append the results to the vectors created at the beginning
    acc(end+1) = actual90;
    CI(end+1) = length90;

end

% Print the results for all T after the loop is done
acc
CI
```

1.2 Output

Table 1 compares the coverage accuracy and the average CI length for different sample sizes (T) at 300 repetitions. Our computers could hardly handle more repetitions than 300 without occupying all computing power for whole days. In this relatively shorter question, we could have used more repetitions, but wanted to stay consistent with all the other questions. It seemed to have no impact, in the sense that the results matched our expectations and a test with more repetitions did not alter the patterns in the outcome, as described in the following paragraph.

As expected, the parametric performs better than the non-parametric bootstrap, because the parametric bootstrap assumes the correct underlying model, namely the regular Student t distribution. Furthermore, both bootstrap model's coverage accuracies increase in the sample size T, whereas the CI lengths decrease in T; this matches our expectations. The actual coverage is very high for the parametric bootstrap, yielding values very close to 1[1]. This is because the parametric is supposed to perform extremely well, as the model assumption therein is correct, matching the true underlying distribution.
The comparison between the Matlab code for the MLE in the book[2] and the built-in

[1]Obviously, this might change if we used more repetitions, but it shows that our code works as intended.

[2]Paolella, M. S. (2018). Fundamental Statistical Inference: A Computational Approach (Vol. 216). John Wiley Sons.

Matlab code shows that both values are nearly identical up to some small simulation error.

Finally, in table 2 we look at different values of bootstrap replications (B) to check whether smaller values than 1000 would have been enough. Since we have shown already in table 1 that the two approaches for calculating the MLE are identical, we only use the function provided in the book for the comparison and for all calculation from here on forth. The reason for this is that the calculations proved to be faster using the function provided in the book. As we can see, the performance (i.e. the actual coverage, but we will call it performance) increases in T. Furthermore, the performance, holding T constant, is very similar for different values of B, showing that B=250 would have already been sufficient.

Correction: We see that the actual coverage of the non-parametric bootstrap increases with the sample size and does converge to the nominal coverage of 90%. Contrary to our expectations, the parametric bootstrap does not perform better. The actual coverage goes far beyond that of the nominal 90% and even seems to converge to 1. This is not, as we initially supposed, an indication of over-achievement, but rather one of bad performance. In the following two subsections, we try to account for different probability levels for the ES and even larger sample sizes. Further, we present the code (not the results) for a double bootstrap, which might also correct for this issue.

Table 1: Parametric and Non-Parametric Bootstrap for True ES based on a Student t Distribution.

	CI Length			Actual Coverage		
	$T = 500$	$T = 1000$	$T = 2000$	$T = 500$	$T = 1000$	$T = 2000$
$df = 4$						
Parametric (MLE Book)	6.6999	4.9340	3.5236	0.9667	0.9867	0.9867
Parametric (MLE Matlab)	6.8780	4.9557	3.5952	0.9667	0.9800	0.9899
Non-Parametric	5.2810	4.1574	3.2968	0.6833	0.7667	0.8133

Note: We use location 1 and scale 2 for the true model. The values were calculated at significance level α=0.1 and for B=1000.

1.3 Using Different Probability Levels for ES

In the following, we try to find a solution for the bad coverage performance we found in reports for the first question. We thought that an actual coverage of close to 1 was an indicator of over-achievement, however, as per your mail, it is simply a bad performance. To amend this, we try two different approaches. Firstly, we use different probability levels for the ES (α)[3] for a constant sample size T and for increasing sample sizes T. As a second approach, we use a double bootstrap, which is 'precisely designed to help correct for this issue'.

[3]Not to be confused with α_stable, which is the tail index for a stable distribution and always has the subscript 'stable'.

Table 2: Effect of B on the Actual Coverage of the Parametric Bootstrap Using the MLE Function Provided in the Book.

	$T = 100$	$T = 500$	$T = 2000$
$B = 250$	0.8810	0.9660	0.9860
$B = 500$	0.8700	0.9590	0.9790
$B = 1000$	0.8840	0.9480	0.9850

Note: We use location 1 and scale 2 for the true model. The values were calculated at significance level α=0.1 for B=250, 500, 1000.

In this section, we use different probability levels for the ES, namely 1%, 5% and 10%. We hope to get an actual coverage around the nominal coverage of 90%, still expecting the parametric bootstrap to perform better than the non-parametric one.

1.3.1 Code

This code is very similar to the one presented in question 1 (1.1), with the exception that we loop over different probability levels for the ES.

```matlab
% ========================Exercise 1. i)============================
% Simulate T-length vectors of IID location scale student t data
% and use parametric and non-parametric !!DOUBLE!! bootstrap to ...
    check whether
% the true ES in the CI. This yields the actual coverage and the
% average lengths of the CI for different proabability levels ...
    for the ES.
% ==================================================================

% Loop over all sample sizes (T) for different probability ...
    levels for the
% ES (alpha_CI):
for T = [1000 10000 100000]
    acc = []; % The output for each T will be appended to these ...
        vectors
    CI = [];

    for alpha_CI = [0.01 0.05 0.1]

    % Define Variables
    rand('twister',6) % set seed value so we can replicate results
    rep = 200;
    p = 0.9; alpha_VaR = 0.01; %alpha_CI = 0.1;
    loc = 1; scale = 2; df = 4; % set parameters for student t ...
        distribution
    initvec = [df loc scale]; % needed for tlikmax0
    B1 = 1e3; ESvec = zeros(B1,1); %set parameters for bootstrap
    bool_acc = zeros(rep,1); length_ci = zeros(rep,1);% vector ...
        for coverage accuracy and CI length


    % Compute True ES to check if it is in CI
    c01 = tinv(alpha_VaR , df); % left tail quantile, for loc-0 ...
        scale-1
    truec = loc+scale*c01; % left tail quantile c
    ES01 = -tpdf(c01,df)/tcdf(c01,df) * (df+c01^2)/(df-1);
    trueES = loc+scale*ES01;


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    % Loop over repetitions
    for i=1:rep

      % Create random variates of student t distribution and ...
          calculate empirical
      % ES for comparison with true ES with bootstrap
      data = loc+scale*trnd(df,T,1); % random variate for ...
          student t distribution

      % Calculation of MLE for parametric bootstrap with matlab ...
          function, output: loc scale df
      %theta_mle_matlab = mle(data,'Distribution','tLocationScale');
      %loc_mle = theta_mle_matlab(1); scale_mle = ...
          theta_mle_matlab(2); df_mle = theta_mle_matlab(3);

      % Calculation of MLE for parametric bootstrap with ...
          function from the book, output: df loc scale
      theta_mle_book = tlikmax0(data, initvec);
      loc_mle = theta_mle_book(2); scale_mle = ...
          theta_mle_book(3); df_mle = theta_mle_book(1);

      % bootstrap
      for b1=1:B1
         %ind = unidrnd(T,[T,1]); bootsamp=data(ind); % nonpara boot
         bootsamp = loc_mle+scale_mle*trnd(df_mle,T,1); % para boot
         VaR = quantile(bootsamp, alpha_VaR);
         temp = bootsamp(bootsamp≤VaR); ESvec(b1)=mean(temp);
      end
      ci = quantile(ESvec,[alpha_CI/2 1-alpha_CI/2]);
      bool_acc(i) = (trueES>ci(1)) & (trueES<ci(2)); % check ...
          wheter the true value is in the CI
      length_ci(i) = ci(2)-ci(1); % calculate the length of the CI

    end
    length90 = mean(length_ci); % calculate mean length of CI
    actual90 = mean(bool_acc); % calculate actual coverage ...
        probability

    acc(end+1) = actual90;
    CI(end+1) = length90;

    end
    disp('====================================================')
    fprintf('T: %d', T)
    acc
    CI

end
```

1.3.2 Output

Since the output in question 1 deviated from our expectations, we tried a new route and used much higher sample sizes ($T = [1'000, 10'000, 100'000]$). We further suspected the performance to be a function of α, the tail probability for the ES. Thus, we extended the results from question 1 to the use of 1%, 5% and 10% and stuck with 300 repetitions for two reasons. Firstly, to stay consistent with the previous results, so they are comparable and secondly, considering the computation time, we decided for higher sample sizes at the cost of lower repetitions. Without these ameliorations, we ended up with an actual coverage that was much higher than the expected nominal coverage of 90% and, similarly, the CI were too long. Incorporating the above changes leads to the reported values in table 3. We can see that the CI length decreases with increasing sample size, and the values for $T = 100'000$ start to look sensible. We do expect the non-parametric to perform worse, so it comes as no surprise that the bigger the sample sizes the closer the values of the CI length for the parametric and non-parametric case. We supposed at the beginning that the bootstrap's performance depends on the tail probability, which can readily be seen by the horizontally declining CI length values. As a premature conclusion, we could say that the including significantly higher sample sizes does have the desired effect. Also, the relationship with the tail probability seems to exist, pointing in the desired direction.
When we turn to the actual coverage, however, the values are all over the place again. We do observe that for the values for the parametric bootstrap seem to tend to 90% when increasing the tail probability, which is what we wanted. This is only an indication and should be explored further by incorporating even higher tail probabilities; we lacked the computational power to obtain more values in such short time. On the other hand, we do not see any relationship with the sample size, which sheds some more light on table 1 and shows that the values there are, thus, also rather constant than increasing. One could even go as far as saying that the actual coverage increases, when looking at values of $\alpha = 0.1$ in table 3.
The performance of the non-parametric shows a clear relationship with the tail probabilities, as expected. However, increasing sample sizes lead to increasing actual coverage beyond the nominal coverage of 90%. To expand on the above conclusion, the dependence on the tail probabilities does seem to exist, however, much lower in magnitude at least for the parametric case. The inclusion of much higher sample sizes, very counter-intuitively, does not lead to better results, which reflects the issues of question 1. If we had more computational power at our demand, we would use more repetitions, however, experiments with lower sample sizes and higher repetitions yielded the same pattern. The natural extension of this is the double bootstrap, which is designed to correct for this issue, as it enhances the accuracy of the single bootstrap[4]. An implementation thereof is given in the next section.

[4]https://digitalcommons.wayne.edu/cgi/viewcontent.cgi?article=1981context=jmasm (15.11.2022).

Table 3: Parametric and Non-Parametric Bootstrap for True ES based on Student t Distribution for Varying Probability Levels for the ES.

	CI Length			Actual Coverage		
	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$
$T = 1'000$						
Parametric	8.9270	5.9291	4.8481	1.0000	0.9933	0.9767
Non-Parametric	16.2329	4.8865	4.3146	0.8800	0.8067	0.7667
$T = 10'000$						
Parametric	2.6301	1.9443	1.6342	1.0000	1.0000	0.9833
Non-Parametric	2.4226	1.9725	1.5721	0.9800	0.9467	0.8867
$T = 100'000$						
Parametric	0.8189	0.6186	0.5183	1.0000	1.0000	0.9967
Non-Parametric	0.8091	0.6084	0.5120	0.9800	0.9633	0.8567

Note: The parametric bootstrap assumes a regular Student t distribution. We use location 0 and scale 1 for the true model with B=1000 and df=4.

1.4 Using Double Bootstrap

In this section, we repeat the procedure of question 1, however, using a double instead of a single bootstrap in order to enhance the coverage performance. We only present the code here, as running a double bootstrap is infeasible. We showed already in question 1 that fewer bootstrap replications (B) are actually sufficient. Since the double bootstrap takes much longer due to the double loop, we should only use $B = 250$ replications, in order to avoid astronomical computation times. In contrast to the previous section, where we used very big sample sizes, we should use the same sample sizes as in the original question (i.e. $T = [500, 1000, 2000]$), to be able to compare the influence of the double to the single bootstrap.

1.4.1 Code

The code looks very similar to the one presented in question 1 (1.1), which the exception of the double bootstrap. The code loops over three parameters: number of repetitions (rep), sample size (T) and the probability level for the ES (alpha_CI). The output is nicely formatted into tables and exported as a .csv file for each number of repetitions. Further, when switching between the parametric and non-parametric, the respective lines have to be commented out *twice*, once in the inner and once in the outer bootstrap. Similarly to question 1, we can still choose the method of MLE.

```
% ====================Exercise 1. ii)=============================
% Simulate T-length vectors of IID location scale student t data
% and use parametric and non-parametric !!DOUBLE!! bootstrap to ...
    check whether
% the true ES in the CI. This yields the actual coverage and the
% average lengths of the CI.
% ================================================================

% Define variables
% Values to loop over
T_vec = [1e2 1e3 1e4 1e5];  len_T = size(T_vec);
alpha_CI_vec = [0.01 0.05 0.1]; len_alpha = size(alpha_CI_vec);
rep_vec = [1e2 1e3 1e4 1e5]; len_rep = size(rep_vec);

truenominal = 0.9; % true nominal coverage
nominal=0.799:0.002:0.999; nomlen=length(nominal);


% Loop over all the variables
for rep = rep_vec
    % Create arrays to append the output to. Do it inside the ...
       loop over the
    % reps to get tables for each number of repetitions
    acc = zeros(len_alpha(2), len_T(2));
    CI = zeros(len_alpha(2), len_T(2));

    for T = T_vec
        iter1 = 1;

        for alpha_CI = alpha_CI_vec
            iter2 = 1;

            % Define Variables
            rand('twister',6) % set seed value so we can ...
                replicate results
            %rep = 1e3;
            alpha_VaR = 0.01; %alpha_CI = 0.1; p = 0.9;
            loc = 1; scale = 2; df = 4; % set parameters for ...
                student t distribution
            initvec = [df loc scale]; % needed for tlikmax0
            B1 = 10; ESvec1 = zeros(B1,1);
            B2 = 10; ESvec2 = zeros(B2,1); %set parameters for ...
                bootstrap
            bool_acc = zeros(rep, 1); length_ci = zeros(rep, 1);
            bool1=zeros(B1, nomlen);


            % Compute True ES to check if it is in CI
            c01 = tinv(alpha_VaR , df); % left tail quantile, ...
                for loc-0 scale-1
            truec = loc+scale*c01; % left tail quantile c
            ES01 = -tpdf(c01,df)/tcdf(c01,df) * (df+c01^2)/(df-1);
            trueES = loc+scale*ES01;


            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


            % Loop over repetitions
            for i=1:rep

              % Create random variates of student t distribution ...
                  and calculate empirical
              % ES for comparison with true ES with bootstrap
              data = loc+scale*trnd(df,T,1); % random variate ...
```

```
                for student t distribution

            % Calculation of MLE for parametric bootstrap with ...
                matlab function, output: loc scale df
            %theta_mle_matlab = ...
                mle(data,'Distribution','tLocationScale');
            %loc_mle = theta_mle_matlab(1); scale_mle = ...
                theta_mle_matlab(2); df_mle = theta_mle_matlab(3);

            % Calculation of MLE for parametric bootstrap with ...
                function from the book, output: df loc scale
            theta_mle_book = tlikmax0(data, initvec);
            loc_mle = theta_mle_book(2); scale_mle = ...
                theta_mle_book(3); df_mle = theta_mle_book(1);

            % bootstrap
            for b1=1:B1 %%%%%%%% outer bootstrap
                %ind = unidrnd(T,[T,1]); bootsamp1=data(ind); % ...
                    nonpara boot
                bootsamp1 = loc_mle+scale_mle*trnd(df_mle,T,1); ...
                    % para boot
                VaR = quantile(bootsamp1, alpha_VaR);
                temp1 = bootsamp1(bootsamp1≤VaR); ...
                    ESvec1(b1)=mean(temp1);

                for b2=1:B2 %%%%%%%% inner bootstrap
                    %ind = unidrnd(T,[T,1]); ...
                        bootsamp2=bootsamp1(ind); % nonpara boot
                    bootsamp2 = ...
                        loc_mle+scale_mle*trnd(df_mle,T,1); % ...
                        para boot
                    VaR = quantile(bootsamp2, alpha_VaR);
                    temp2 = bootsamp2(bootsamp2≤VaR); ...
                        ESvec2(b2)=mean(temp2);
                end  %%%%%% end inner bootstrap
                for j = 1:nomlen
                   alpha_inner=1-nominal(j) ; ci=quantile ...
                       (ESvec2 , [alpha_inner/2 ...
                       1-alpha_inner/2]) ;
                   bool1(b1,j) = (trueES>ci(1)) & (trueES<ci(2));
                end

             end %%%%%%% end outer bootstrap

             bootactual = mean(bool1) + cumsum((1:nomlen) / ...
                 1e10 );
             boot90=interp1(bootactual, nominal, truenominal);
             alphanom=1-boot90; ci1 = quantile(ESvec1 ...
                 ,[alphanom/2 1-alphanom/2]);
             bool_acc(i) = (trueES>ci1(1)) & (trueES<ci1(2)); ...
                 % check wheter the true value is in the CI
             length_ci(i) = ci1(2)-ci1(1); % calculate the ...
                 length of the CI

         end
         length90 = mean(length_ci); % calculate mean length ...
             of CI
         actual90 = mean(bool_acc); % calculate actual ...
```

```
                    coverage probability

                % Append output to tables
                acc(iter2, iter1) = length90;
                CI(iter2, iter1) = actual90;

                iter2 = iter2 + 1;
            end

            iter1 = iter1 + 1;
        end
        % Print the number of repetitions and the output tables
        fprintf('Table for %f repetitions. \nColumns are sample ...
           sizes, rows are probability levels for ES.\n', rep)

        % Create tables from output and display them
        tab_acc = array2table(acc, 'VariableNames', string(T_vec), ...
           'RowNames', string(alpha_CI_vec))
        tab_CI = array2table(CI, 'VariableNames', string(T_vec), ...
           'RowNames', string(alpha_CI_vec))

        % Write output to csv file for each number of repetitions
        current_file_acc = sprintf('acc__reps_%d.csv', rep);
        current_file_CI = sprintf('CI__reps_%d.csv', rep);
        writetable(tab_acc, current_file_acc, 'WriteVariableNames', ...
           true, 'WriteRowNames', true, 'Delimiter', ',');
        writetable(tab_CI, current_file_CI, 'WriteVariableNames', ...
           true, 'WriteRowNames', true, 'Delimiter', ',');

        disp('==========================================================')
    end
```

2 Parametric and Non-Parametric Bootstrap of ES based on Noncentral t Distribution (Student t Assumption in Param. Bootstrap)

In question 2 we first compare the outputs of the density approximations given in the book to the one built-into Matlab. We find that they are identical. Then, we compare the calculation of the ES via the integral definition of the Noncentral t Distribution (NCT) and via simulation. We further report the average length of the CIs, and the actual coverage using parametric and non-parametric bootstrap. The twist here, compared to question 1, is that the parametric bootstrap does *not* assume the correct model. It assumes a central Student t distribution, whereas the underlying data is simulated from a NCT distribution.

In the following questions, we will use the singly NCT distribution, where μ represents the noncentrality parameter and df the degrees of freedom. The central case of $\mu = 0$ results in the regular or central Student t distribution, whilst negative values of μ lead to heavier left tails of the distribution (and heavier right tails in the case of a positive μ). The former case is more relevant in the context of real financial stock returns, so we will focus on negative values of μ.

2.1 Comparison PDF from Book and Matlab

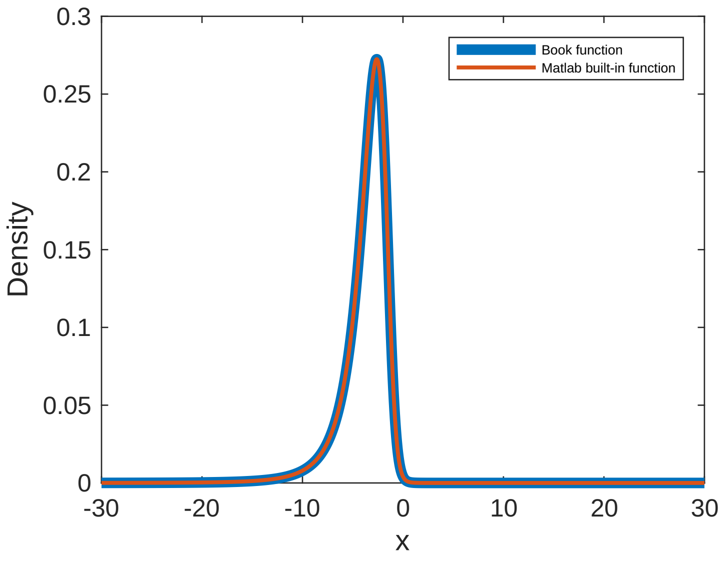
For calculating the ES via the integral definition of the NCT distribution, we need to compare the density function from the book Fundamental Statistical Inference[5] and the Matlab built-in density function. In figure 1 we see that they both give the same values. The density approximation from the book can be found in the Appendix (5.2). It calculates the log of the density, which is why we apply the exponential function to make it comparable to the built-in Matlab function.

[5]Paolella, M. S. (2018). Fundamental Statistical Inference: A Computational Approach (Vol. 216). John Wiley Sons.

2.1.1 Code and Output

```
% ========================Exercise 2.i)============================
% Comparison of the book matlab code and the matlab built-in ...
    function
%==================================================================

% Define Variables and x-axis
df = 4; mu = -3;
x = linspace(-30,30,10000);

% Approximate densities
f = exp(stdnctpdfln_j(x, df, mu)); % book -> exp to reverse the log
f2 = nctpdf(x,df, mu); % Matlab

% Graph the two densities on the same figure
figure
plot(x, f, 'LineWidth', 4)
hold on
plot(x, f2, 'Linewidth', 1.5)
xlabel('x')
ylabel('Density')
l=legend('Book function','Matlab bilt-in ...
    function','Location','northeast')
set(l,'FontSize',5.5);
```

Figure 1: Comparison of PDF using the Book and Matlab Function.



2.2 Expected Shortfall via Simulation and with Numeric Integration

We compute the true ES of a NCT distribution in two ways. Firstly via simulation and secondly by using the integral definition of the NCT, which means we first solve for the Value-at-Risk (VaR) quantile[6] and then do numeric integration of the NCT density multiplied by x. In figure 2 we can see that the ES with numeric integration appears to be around the mean of all ES via simulation, which is what we were hoping for.

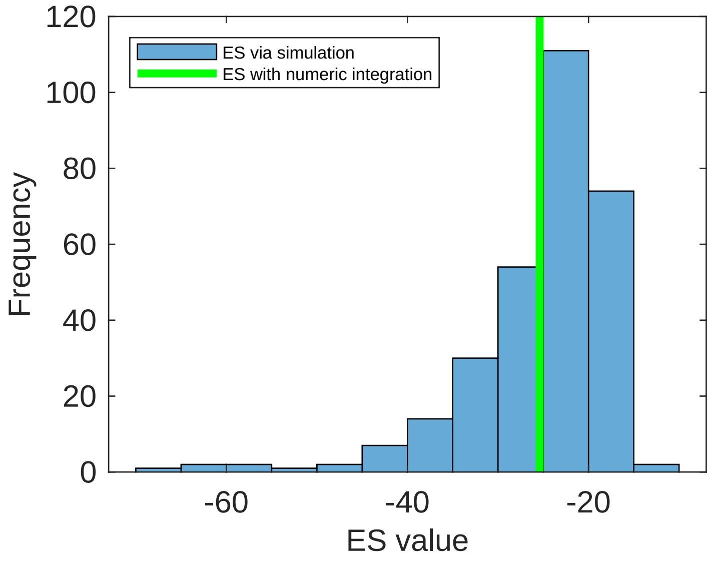
2.2.1 Code and Output

```
% ========================Exercise 2.ii)===========================
% Comparison of the expected shortfall via simulation and with ...
    numeric integration
% ================================================================

% Define Variables
rep = 300; T = 500;
p = 0.9; alpha_CI = 0.1; alpha_VaR = 0.01;
loc = 0; scale = 1; df = 3; % set parameters for student t ...
    distribution
mu = -3; % mu is noncentrality parameter

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ES with numeric integration:
c01 = nctinv(alpha_VaR,df,mu); % left tail quantile
I01 = @(x) x.*exp(stdnctpdfln_j(x,df,mu)); %book pdf function
ES_01_numint = integral(I01 , -Inf , c01) / alpha_VaR; %upper ...
    bound is the alpha-quantile
cLS = loc+scale*c01; % cLS is cutoff Location Scale
ILS = @(y) (y).*exp(stdnctpdfln_j((y-loc)/scale, df,mu))/scale;
ES_wLS_numint = integral(ILS , -Inf , cLS) / alpha_VaR

% ES via simulation
ESvec=zeros(rep,1);
for i=1:rep
  data=loc+scale*nctrand(df,mu,T); VaR=quantile (data, alpha_VaR);
  temp=data(data≤VaR); ESvec(i)=mean(temp);
end
ES_via_simulation=mean(ESvec)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Graph of ES via simulation and ES with numeric integration :
figure
histogram(ESvec), ax=axis;
set(gca,'fontsize',12)
line ([ ES_wLS_numint ES_wLS_numint] ,[0 ax(4)], 'color', 'g ', ...
    'linewidth',3)
xlabel('ES value','FontSize',12)
```

[6]Also simply referred to as the α quantile. But in order to avoid confusion with the tail index α of a stable Paretian distribution used in question 3, we use VaR-quantile.

```
ylabel('Frequency','FontSize',12)
l=legend('ES via simulation','ES with numeric ...
    integration','Location','northwest')
set(l,'FontSize',5);
```

Figure 2: Comparison of the ES via Simulation and with Numeric Integration.



2.3 Main Part

2.3.1 Code

The function for the MLE of a regular Student t distribution (5.3), the function for the simulation on NCT random variates (5.1) and lastly the function provided in the book for the calculation of the log of the pdf for a NCT distribution (5.2) can be found in the Appendix. Again, the code is set up so that it iterates over all possible values of the sample size (T), the degrees of freedom (df) and the noncentrality parameter (μ). To switch between the parametric and non-parametric bootstrap, comment out the respective lines (64 or 66). As discussed in question 1 (1.1), we use the simple function provided in the book for the calculation of the MLE for the regular Student t distribution in the parametric bootstrap. For completeness, the more sophisticated MLE function for the regular Student t is provided in the Appendix (5.5).

```
% ========================Exercise 2=============================
% Simulate T-length vectors of IID noncentral t random variates for
% calculation of true ES and check the CI based on parametric
% (using regular Student t distribution) and non-parametric ...
    bootstrap.
%==================================================================

% We have 3 for loops. We use different values for the sample ...
    size T, df,
% and the noncentrality parameter mu.

```

```
for mu = [-3 -2 -1 0]
    for df = [3 6]
        acc = []; % The output for each T will be appended to
        % these vectors
        CI = [];
        for T = [100 500 2000]

            % Define Variables
            rand('twister',6) % set seed value so we can
            %replicate results
            rep=200;
            p = 0.9; alpha_CI = 0.1; alpha_VaR = 0.01;
            loc = 0; scale = 1; % df = 3 set parameters for ...
                student t
            % distribution
            % mu = -3; % mu is noncentrality parameter
            initvec = [df loc scale]; % needed for tlikmax0
            B1 = 1e3; ESvec = zeros(B1,1); %set parameters for
            % bootstrap
            bool_acc = zeros(rep,1); length_ci = zeros(rep,1);
            % vector for coverage accuracy and CI length

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


            %ES with numeric integration:
            c01 = nctinv(alpha_VaR,df,mu); % left tail quantile
            %I01 = @(x) x.*nctpdf(x,df,mu); %matlab pdf function
            I01 = @(x) x.*exp(stdnctpdfln_j(x,df,mu)); %book pdf ...
                function
            ES_01_numint = integral(I01 , -Inf , c01) / ...
                alpha_VaR; %upper bound is the alpha-quantile


            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


            % Loop over repetitions
            for i=1:rep

              ESvec = zeros(B1,1);
              %i % to see at which iteration you are

              % Create random variates of student t distribution
              % and calculate empirical
              % ES for comparison with true ES with bootstrap
              data = loc + scale*nctrand(df, mu, T); % random
              % variate for student t distribution

              % Calculation of MLE for parametric bootstrap with
              %function from the book, output: df loc scale
              theta_mle_book = tlikmax0(data, initvec);
              loc_mle = theta_mle_book(2); scale_mle = ...
                  theta_mle_book(3); df_mle = theta_mle_book(1);

              % bootstrap
              for b1=1:B1
                  %ind = unidrnd(T,[T,1]); bootsamp=data(ind);
                  % nonpara boot
                  bootsamp = loc_mle+scale_mle*trnd(df_mle,T,1);
                  % para boot
                  VaR = quantile(bootsamp, alpha_VaR);
                  temp = bootsamp(bootsamp≤VaR); ...
                     ESvec(b1)=mean(temp);
              end
              ci = quantile(ESvec,[alpha_CI/2 1-alpha_CI/2]);
              bool_acc(i) = (ES_01_numint>ci(1)) & ...
                  (ES_01_numint<ci(2));
              % check whether the true value is in the CI
              length_ci(i) = ci(2)-ci(1);
              % calculate the length of the CI
            end
            length90 = mean(length_ci); % calculate mean length
            % of CI
            actual90 = mean(bool_acc); % calculate actual
            %coverage probability

            % Append the results to the result vectors
            acc(end+1) = actual90;
            CI(end+1) = length90;
        end
        disp('===================================================')

        % The according parameters are printed out before every
        % output,
        % so it is easier to copy them to the Latex file.
        fprintf('mu: %d, df: %d\n', mu, df)
        acc
        CI
    end
    disp('=======================================================')
end
disp('===========================================================')
```

2.3.2 Output

Tables 4-7 represent our results for the parametric and non-parametric bootstrap of the true ES based on a NCT distribution with different values for μ ranging from -3 to 0. The crucial part of this question is that we assume a regular Student t distribution in the parametric bootstrap, which is incorrect. Followingly, we expect the parametric to perform much worse than the non-parametric, which only resamples from the original dataset, i.e. the NCT random variates. We use 300 repetitions in our calculations.

The actual coverage increase whilst the average CI length decrease for the non-parametric bootstrap with increasing sample size T, which is exactly what we expect. Namely, the more samples we have, the more concise is the sampling distribution of the ES. Thus, we should be able to have higher actual coverage of the true ES for an ever smaller CI.
The non-parametric values do not vary with a change in the noncentrality parameter μ. This makes sense, as the non-parametric simply resamples from the NCT random variates that change according to the value of μ and does not falsely assume

a central distribution. Using similar logic, we do not expect the performance of the non-parametric to be influenced by a change in the number of df in the underlying NCT distribution.

On the other hand, if we look at the values for the parametric bootstrap, we see, as expected, a much poorer performance. This is due to the false assumption that the underlying model is a regular Student t distribution, which it is not. This would imply that the bigger the noncentrality parameter (μ) in absolute value, the worse the assumption of a regular Student t. Or the other way around, if we have $\mu = 0$, the parametric should outperform the non-parametric, because it assumes the correct model, namely a central Student t distribution. This is exactly what we see in table 7. The actual coverage for the parametric explodes and lies much above the one for the non-parametric. Similarly, an increase in sample size leads to even higher actual coverage and decreasing CI length. Our initial expectations are verified.

Now, we turn to the performance of the parametric bootstrap in the case where μ is smaller than 0, i.e. there is some noncentrality. Under these conditions, we would expect the model to perform worse with increasing sample size. This is because the sample size would decrease the CI length for a central Student t, i.e. make it more narrow around the parameter under the false centrality-assumption, which is not the true parameter under the NCT distribution. Followingly, the true parameter lies outside the bounds of the CI more often. Obviously, we still expect a smaller CI length with increasing sample size. We report values for the parametric bootstrap with $\mu < 0$ in tables 4-6, which support our arguments.
We also see that the CI lengths are much shorter for higher df. This is because higher df result in lighter tails, which means that the empirical ES is calculated using a shorter range on the x-axis, i.e. the values from negative infinity to the VaR-quantile. Therefore, the values for the empirical ES in the bootstrap lie closer to each other and their CI is shorter by construction.

The asymmetry of the NCT distribution will become smaller the larger the df. Therefore, we expect the parametric model, which assumes centrality, to perform better with higher df. Similarly, the parametric bootstrap should perform better the closer μ gets to 0, i.e. the less asymmetry there is in the true distribution. However, we do not observe the last two claims in the data. Even after days worth of reviewing the code, checking with classmates and running countless variations of the code, we could not get rid of these inconsistencies. Given that the *exact* same code works fine in the other question, we arrived at three possible explanations for this. **Firstly**, "the usual skewness is not generally a good measure of asymmetry for this distribution, because if the degrees of freedom is not larger than 3, the third moment does not exist at all. Even if the degrees of freedom is greater than 3, the sample estimate of the skewness is still very unstable unless the sample size is very large."[7] This leads us to the conclusion that even though we expected the noncentrality parameter μ to shift the underlying NCT distribution to the left, resulting in heavier left tails, the estimate of the skewness used in the MLE for the regular t distribution in the parametric bootstrap might suffer. We tried to verify this explanation by running the same calculations for higher df and larger sample sizes, without finding

[7]https://en.wikipedia.org/wiki/Noncentral_t-distribution (10.11.2022).

a different pattern in the outcomes. Therefore, we do not report all the tables again. **Secondly**, maybe this problem fades away if we used more repetitions in the calculations. We checked this as well, running all calculations with more repetitions up to 100'000, again without much luck. We did not, however, check through all specifications with more repetitions, because our computers would have been occupied for far too long, rendering all other work cumbersomely slow. **Thirdly**, as discussed in question 1 (1.1), we initially did not use the more sophisticated MLE function from the book, which might lead to weaker parameter estimates. This, however, should have caused problems in the other questions as well, which it did not. For the sake of completeness, we did run the whole specifications using the improved version, without much luck either. Since the inconsistencies remained, we refrained from reporting all tables again.
In conclusion, our code yields the expected results in all other question and also in this one, except for the dependence of the actual coverage the μ and the df. We have gone to great lengths to try to find a solution to this, but we did not succeed. To quote Arthur Conan Doyle's Sherlock Holmes: "When you have eliminated all which is impossible, then whatever remains, however improbable, must be the truth." So, maybe there just is no such dependence, or it shows only under even bigger sample sizes and repetitions.

Table 4: Parametric and Non-Parametric Bootstrap for True ES based on a Non-central Student t Distribution with for $\mu = -3$.

	CI Length			Actual Coverage		
	$T = 500$	$T = 1000$	$T = 2000$	$T = 500$	$T = 1000$	$T = 2000$
$df = 3$						
Parametric	21.6540	16.6287	12.2271	0.8000	0.8400	0.7100
Non-Parametric	15.2096	12.5672	10.9797	0.6633	0.7600	0.8267
$df = 6$						
Parametric	4.0480	2.8728	2.1209	0.5767	0.3433	0.0900
Non-Parametric	3.5955	2.9667	2.2447	0.6833	0.8167	0.8267

Note: The parametric bootstrap assumes a regular Student t distribution. We use location 0 and scale 1 for the true model with B=1000.

Table 5: Parametric and Non-Parametric Bootstrap for True ES based on a Non-central Student t Distribution with for $\mu = -2$.

	CI Length			Actual Coverage		
	$T = 500$	$T = 1000$	$T = 2000$	$T = 500$	$T = 1000$	$T = 2000$
$df = 3$						
Parametric	14.0067	10.3630	7.6569	0.7667	0.7133	0.5100
Non-Parametric	12.1644	9.4797	7.4663	0.6633	0.7133	0.8033
$df = 6$						
Parametric	2.9615	2.1443	1.5592	0.5667	0.3467	0.0933
Non-Parametric	3.1014	2.3581	1.7901	0.7233	0.7967	0.8400

Note: The parametric bootstrap assumes a regular Student t distribution. We use location 0 and scale 1 for the true model with B=1000.

Table 6: Parametric and Non-Parametric Bootstrap for True ES based on a Non-central Student t Distribution with for $\mu = -1$.

	CI Length			Actual Coverage		
	$T = 500$	$T = 1000$	$T = 2000$	$T = 500$	$T = 1000$	$T = 2000$
$df = 3$						
Parametric	8.0673	5.9555	4.4706	0.6933	0.6167	0.4200
Non-Parametric	7.5219	5.9842	5.0152	0.6633	0.7067	0.8067
$df = 6$						
Parametric	2.2438	1.6656	1.1667	0.6633	0.5967	0.2767
Non-Parametric	2.1405	1.6900	1.2977	0.6767	0.7900	0.8567

Note: The parametric bootstrap assumes a regular Student t distribution. We use location 0 and scale 1 for the true model with B=1000.

Table 7: Parametric and Non-Parametric Bootstrap for True ES based on a Non-central Student t Distribution with for $\mu = 0$.

	CI Length			Actual Coverage		
	$T = 500$	$T = 1000$	$T = 2000$	$T = 500$	$T = 1000$	$T = 2000$
$df = 3$						
Parametric	6.0242	4.5187	3.2884	0.9633	0.9933	0.9900
Non-Parametric	4.8600	3.5308	2.8819	0.6667	0.7133	0.7933
$df = 6$						
Parametric	1.9296	1.3804	1.0012	0.9433	0.9667	0.9833
Non-Parametric	1.5595	1.2474	0.9115	0.6933	0.8100	0.8367

Note: The parametric bootstrap assumes a regular Student t distribution. We use location 0 and scale 1 for the true model with B=1000.

3 Parametric and Non-Parametric Bootstrap of ES based on Stable Paretian Distribution (Student t Assumption in Param. Bootstrap)

In question 3 we repeat what we did in question 2, but we are now using the symmetric stable Paretian as the underlying distribution with different tail indices α, as opposed to different df. We then calculate the true ES using the results from Stoyanov et al.[8]. We then construct confidence intervals for the empirical ES, based on the parametric bootstrap (wrongly assuming the Student t distribution), and the non-parametric bootstrap to check the actual coverage and the average CI lengths.

3.1 Code

Again, the code is set up so that it iterates over all possible specifications. To switch from between the parametric and non-parametric bootstrap, comment out the respective lines (51 or 53). As before, we use the simple function provided in the book to calculate the MLE for a regular Student t distribution in the parametric bootstrap, which can be found in the Appendix (5.3). For completeness, the more sophisticated MLE function for the regular Student t is provided in the Appendix (5.5).

```
 1  % ========================Exercise 3==============================
 2  % Simulate T-length vectors of IID stable paretian random ...
      variates for
 3  % calculation of true ES and check the CI based on parametric
 4  % (using regular Student t distribution) and non-parametric ...
      bootstrap.
 5  % ===============================================================
 6  
 7  % We have 3 for loops. We use different values for the sample
 8  % size T, tail index alpha_stable,
 9  % and the noncentrality parameter mu.
10  
11  for alpha_stable = [1.6 1.8]
12          acc = []; % The output for each T will be appended to
13          % these vectors
14          CI = [];
15          for T = [500 1000 2000]
16  
17              % Define Variables
18              rand('twister',6) % set seed value so we can
19              % replicate results
20              rep = 300; %T = 2000;
21              p = 0.9; alpha_CI = 0.1; alpha_VaR = 0.01;
22              %loc = 0; scale = 1; %df = 6; % set parameters for
23              % student t distribution
24              %mu = -3; % mu is noncentrality parameter
25              %alpha_stable = 1.8;
```

[8]Stoyanov, S. V., Samorodnitsky, G., Rachev, S., Ortobelli Lozza, S. (2006). Computing the portfolio conditional value-at-risk in the α-stable case. Probability and Mathematical Statistics, 26(1), 1-22.

```
            skewness_stable = 0; scale_stable = 1; mu_stable = ...
                0; % parameters for stable distribution
            initvec = [df loc scale]; % needed for tlikmax0
            B1 = 1e3; ESvec = zeros(B1,1); %set parameters for
            % bootstrap
            bool_acc = zeros(rep,1); length_ci = zeros(rep,1);
            % vector for coverage accuracy and CI length

            % Compute True ES to check if it is in CI
            data_ES = stblrnd(alpha_stable, skewness_stable, ...
                scale_stable, mu_stable, T, 1);
            q=quantile(data_ES, alpha_VaR);
            trueES = (scale_stable*Stoy(q, alpha_stable, ...
                skewness_stable)/alpha_VaR)+mu_stable;

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            % Loop over repetitions
            for i=1:rep
              % Create random variates of student t distribution
              % and calculate empirical
              % ES for comparison with true ES with bootstrap
              data = stblrnd(alpha_stable, skewness_stable, ...
                  scale_stable, mu_stable, T, 1); % random ...
                  variate for student t distribution

              % Calculation of MLE for parametric bootstrap with
              % function from the book, output: df loc scale
              theta_mle_book = tlikmax0(data, initvec);
              loc_mle = theta_mle_book(2); scale_mle = ...
                  theta_mle_book(3); df_mle = theta_mle_book(1);

              % bootstrap
              for b1=1:B1
                 ind = unidrnd(T,[T,1]); bootsamp=data(ind);
                 % nonpara boot
                 %bootsamp = loc_mle+scale_mle*trnd(df_mle,T,1);
                 % para boot
                 VaR = quantile(bootsamp, alpha_VaR);
                 temp = bootsamp(bootsamp≤VaR); ...
                     ESvec(b1)=mean(temp);
              end
              ci = quantile(ESvec,[alpha_CI/2 1-alpha_CI/2]);
              bool_acc(i) = (trueES>ci(1)) & (trueES<ci(2));
              % check whether the true value is in the CI
              length_ci(i) = ci(2)-ci(1); % calculate the length ...
                  of the CI

            end
            length90 = mean(length_ci); % calculate mean length
            % of CI
            actual90 = mean(bool_acc); % calculate actual
            % coverage probability

            acc(end+1) = actual90;
            CI(end+1) = length90;

        end
        disp('=====================================================')

        % The according parameters are printed out before every
        % output,
        % so it is easier to copy them to the Latex file.
        fprintf('alpha_stable: %d\n', alpha_stable)
        acc
        CI
end
disp('==========================================================')
```

3.2 Output

Table 8 report our results for the parametric and non-parametric bootstrap for a stable Paretian distribution, while the parametric bootstrap wrongly assumes a regular Student t distribution. We run all the calculations twice for different values of the tail index α. We again use 300 repetitions.

We start by looking at the values for the non-parametric bootstrap. We assume that it outperforms the parametric one, as it simply resamples for the simulated stable Paretian random variates. As in question 2, we do not expect the non-parametric values to change much with different model specifications, as it is based on resampling. Table 8 shows that the non-parametric performs okay. The actual coverage increases with the sample size for the non-parametric case, and the CI lengths for both bootstraps decrease with increasing sample size.
We expect the parametric model to perform very poorly, which can be seen from the very bad actual coverage values.

However, we would expect the parametric model's performance to depend on the tail index α of the underlying stable Paretian distribution. Similarly to what we expected in question 2 with the dependence on the noncentrality parameter μ. It is really hard to see concrete evidence for a difference between the coverage probabilities. This would have posed as an opportunity to shed light on the inconsistency observed earlier, however, it is not of much help either. We did try other tail indices, but they also did not help at all in the explanation of the missing dependence.
The CI lengths, however, are considerable smaller for a bigger tail index. This might be because the tail index describes the rate at which the tails taper off. The closer α gets from 1.6 (the Cauchy distribution) to 2, the closer the stable distribution gets to the normal distribution. This means that the estimate for the df of a regular Student t tends to infinity in the MLE, i.e. the estimated df in the parametric bootstrap are very high. Since the tails of a distribution with higher df, or the normal distribution in the limit, have much lighter tails, the empirical ES uses a much shorter range of values (on the x-axis: that is the x-values from negative infinity to the specified VaR-quantlile.) for its calculation. Therefore, the values lie closer to each other, hence, the CI are shorter by construction. Therefore, we have established the same relationship of the df and the size of the CI lengths that we were observed in question 2 (2.3.2).

Table 8: Parametric and Non-Parametric Bootstrap for True ES based on a Stable Paretian Distribution with Different Tail Indices α.

	CI Length			Actual Coverage		
	$T = 500$	$T = 1000$	$T = 2000$	$T = 500$	$T = 1000$	$T = 2000$
$\alpha = 1.6$						
Parametric	10.1968	7.7960	5.5838	0.6567	0.2467	0.0367
Non-Parametric	18.5696	18.6679	13.3921	0.5700	0.5800	0.6100
$\alpha = 1.8$						
Parametric	3.8102	2.8005	1.9589	0.8267	0.1867	0.0033
Non-Parametric	7.5772	7.6816	5.4951	0.6767	0.5933	0.5533

Note: The stable distribution has a skewness and location of 0 and a scale of 1 for simplicity. We still use B=1000.

4 Parametric and Non-Parametric Bootstrap of ES based on Noncentral t Distribution (Noncentral t Assumption in Param. Bootstrap)

In question 4 we repeat the procedure of question 1, however, using the NCT instead of the regular Student t for the data generating process (DGP) and as the model assumption in the parametric bootstrap. Again, we calculate a 90% bootstrap confidence interval based on B bootstrap replications and report the actual coverage and also the CI lengths, as a functions of T, for both the parametric and non-parametric bootstrap. Further, we compare the computation time of the built-in Matlab approximation for the pdf of the NCT to the "d.d.a" NCT approximation from the book[9] in the MLE of the location-scale NCT's parameters.

4.1 Code

The function for the calculation of the simple and more sophisticated MLE parameters of a NCT distribution (5.4, resp. 5.6) and the function provided in the statistical inference book for the density approximation of a NCT distribution (5.2) are listed in the Appendix. Again, the code is set up so that it iterates over all specifications. In order to change from parametric to non-parametric bootstrap, comment out the respective lines (58 or 59). Depending on what density approximation should be used in the MLE, comment out the respective lines (52 or 53). The simple MLE function that we used in the previous questions was readily adapted to the NCT by adding the noncentrality parameter and replacing the pdf for the central Student t with the pdf approximations of the NCT distribution. Obviously, the same can be done for the more sophisticated MLE function provided in the book, paying attention to its output, which also consists of the standard errors for the estimates.

[9]Paolella, M. S. (2018). Fundamental Statistical Inference: A Computational Approach (Vol. 216). John Wiley Sons.

```
% =========================Exercise 4================================
% Simulate T-length vectors of IID noncentral t random variates for
% calculation of true ES and check the CI based on parametric
% (also nct distribution) and non-parametric bootstrap.
%====================================================================

% Start the timer for the elapsed time for the whole run
tic

% We have 3 for loops. We use different values for the sample
% size T, df,
% and the noncentrality parameter mu.

for mu = [-3 -2 -1 0]
    for df = [3 6]
        acc = []; % The output for each T will be appended to
        %these vectors
        CI = [];
        for T = [500 1000 2000]

            % Define Variables
            rand('twister',6) % set seed value so we can
            % replicate results
            rep = 300; %T = 250;
            p = 0.9; alpha_CI = 0.1; alpha_VaR = 0.01;
            loc = 0; scale = 1; % set parameters for student t
            % distribution
            initvec = [df mu loc scale]; % for nctlikmax estimation
            B1 = 1e3; ESvec = zeros(B1,1); %set parameters for ...
                bootstrap
            bool_acc = zeros(rep,1); length_ci = zeros(rep,1);
            % vector for coverage accuracy and CI length

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %ES with numeric integration:
            c01 = nctinv(alpha_VaR,df,mu); % left tail quantile
            %I01 = @(x) x.*nctpdf(x,df,mu); %matlab pdf function
            I01 = @(x) x.*exp(stdnctpdfln_j(x,df,mu)); %book pdf ...
                function
            ES_01_numint = integral(I01 , -Inf , c01) / ...
                alpha_VaR; %upper bound is the alpha-quantile

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            % Loop over repetitions
            for i=1:rep
              % Create random variates of student t distribution ...
                  and calculate empirical
              % ES for comparison with true ES with bootstrap
              data = loc + scale*nctrand(df, mu, T); % random
              % variate for student t distribution

              % Calculation of MLE for parametric bootstrap with
              % matlab/book pdf
              %param_mle = nctlikmax_matlab(data, initvec);
              param_mle = nctlikmax_book(data, initvec);
              df_mle = param_mle(1); mu_mle = param_mle(2); ...
                 loc_mle = param_mle(3); scale_mle = param_mle(4);

              % bootstrap
              for b1=1:B1
                 ind = unidrnd(T,[T,1]); bootsamp=data(ind); % ...
                    nonpara boot
                 %bootsamp = loc_mle+scale_mle*nctrand(df_mle, ...
                    mu_mle, T); % para boot
                 VaR = quantile(bootsamp, alpha_VaR);
                 temp = bootsamp(bootsamp≤VaR); ...
                    ESvec(b1)=mean(temp);
              end
              ci = quantile(ESvec,[alpha_CI/2 1-alpha_CI/2]);
              bool_acc(i) = (ES_01_numint>ci(1)) & ...
                 (ES_01_numint<ci(2)); % check whether the true ...
                 value is in the CI
              length_ci(i) = ci(2)-ci(1); % calculate the length
              % of the CI

            end
            length90 = mean(length_ci); % calculate mean length
            %of CI
            actual90 = mean(bool_acc); % calculate actual
            % coverage probability

            acc(end+1) = actual90;
            CI(end+1) = length90;
        end
        disp('====================================================')

        % The according parameters are printed out before every
        % output,
        % so it is easier to copy them to the Latex file.
        fprintf('mu: %d, df: %d\n', mu, df)
        acc
        CI
    end
    disp('========================================================')
end
disp('============================================================')


% End the timer
toc
```

4.2 Output

The code computes the parametric and non-parametric bootstrap for a NCT distribution, correctly assuming the true model is a NCT distribution in the parametric bootstrap. In order to estimate the parameters for the parametric bootstrap, we calculate the MLE based on the density approximation provided in the book and the one built into Matlab. We run both versions for one repetition of one specification and without going into the details of the parameter specification of this comparison run, we see that the calculation based on the pdf provided in the book is about 1.4 times faster. This result seems to hold roughly constant over different specifications, so we only use the density approximation from the book for the outputs reported in tables 9-12. Again, we use 300 repetitions.

Since the parametric bootstrap assumes the correct model this time, we expect the performance of the parametric to be much higher than that of the non-parametric. Indeed, tables 9-12 show exactly this. We can also see that for both parametric and non-parametric, the actual coverage increases and the CI length decreases with increasing sample size, as was to be expected. As before, we do not expect the performance of the non-parametric to change with different model specification, as it is based solely on resampling; the reported values support this.
The performance of the parametric bootstrap also does not change much with different parameter specifications, which makes sense, because it always assumes the correct model and gets the 'correct' MLEs for each parameter in the specification. This is in contrast to question 2, where the parametric performance was expected to be dependent on the NCT model parameters, which specified how far off the centrality-assumption of the parametric bootstrap really is compared to the NCT distribution. Lastly, we see again that higher df lead to shorter CIs, which follows the same logic as explained in question 2 (2.3.2).

Correction: The same correction as in question 1 is appropriate here. The non-parametric bootstrap seems to converge to the nominal coverage of 90% as expected, however, the parametric one exceeds it by far. Using the same logic and approach as in the extension of question 1, we would check for different probability levels for the ES and for bigger sample sizes. On top of that, we could also use a double bootstrap, see 1.4 for the implementation.

Table 9: Parametric and Non-Parametric Bootstrap for True ES based on a Non-central Student t Distribution with for $\mu = -3$.

	CI Length			Actual Coverage		
	$T = 100$	$T = 500$	$T = 2000$	$T = 100$	$T = 500$	$T = 2000$
$df = 3$						
Parametric	22.0855	16.0968	11.7550	0.9667	0.9900	0.9900
Non-Parametric	14.8778	12.3103	9.8671	0.6400	0.7133	0.8200
$df = 6$						
Parametric	4.8423	3.4287	2.4442	0.9733	0.9767	0.9867
Non-Parametric	3.8775	2.9579	2.3311	0.7100	0.7567	0.8533

Note: The parametric bootstrap also assumes a NCT distribution. We use location 0 and scale 1 for the true model with B=1000.

Table 10: Parametric and Non-Parametric Bootstrap for True ES based on a Noncentral Student t Distribution with for $\mu = -2$.

	CI Length			Actual Coverage		
	$T = 100$	$T = 500$	$T = 2000$	$T = 100$	$T = 500$	$T = 2000$
$df = 3$						
Parametric	15.4827	11.6884	8.5130	0.9500	0.9733	0.9967
Non-Parametric	10.8619	9.3950	7.3677	0.6700	0.7267	0.8000
$df = 6$						
Parametric	3.6364	2.6640	1.9387	0.9300	0.9500	0.9900
Non-Parametric	3.0504	2.3592	1.7219	0.7100	0.7900	0.8133

Note: We use location 0 and scale 1 for the true model with B=1000.

Table 11: Parametric and Non-Parametric Bootstrap for True ES based on a Noncentral Student t Distribution with for $\mu = -1$.

	CI Length			Actual Coverage		
	$T = 100$	$T = 500$	$T = 2000$	$T = 100$	$T = 500$	$T = 2000$
$df = 3$						
Parametric	10.2858	7.4758	5.6105	0.9500	0.9767	0.9867
Non-Parametric	7.6267	6.3333	4.9702	0.6800	0.7233	0.7767
$df = 6$						
Parametric	2.6903	1.9641	1.3981	0.9267	0.9533	0.9633
Non-Parametric	2.2297	1.7143	1.2949	0.7233	0.7567	0.8400

Note: We use location 0 and scale 1 for the true model with B=1000.

Table 12: Parametric and Non-Parametric Bootstrap for True ES based on a Noncentral Student t Distribution with for $\mu = -0$.

	CI Length			Actual Coverage		
	$T = 100$	$T = 500$	$T = 2000$	$T = 100$	$T = 500$	$T = 2000$
$df = 3$						
Parametric	6.0734	4.4881	3.3044	0.9833	0.9767	0.9900
Non-Parametric	4.3268	3.7999	2.8948	0.6433	0.7567	0.8167
$df = 6$						
Parametric	1.9044	1.3674	0.9749	0.9167	0.9533	0.9467
Non-Parametric	1.5283	1.2064	0.9417	0.6567	0.7967	0.8167

Note: We use location 0 and scale 1 for the true model with B=1000.

5 Appendix: Functions

5.1 Simulation of Noncentral t Random Variates

```
% Simulates T random variates from a nct distribution with ...
    degrees of
% freedom (df) and noncentrality parameter mu.
% We specifically do not use the name 'nctrnd' because this function
% already exists in Matlab.
function NCT = nctrand(df, mu, T)
N = normrnd(mu,1,T,1);
X = chi2rnd(df,T,1);
NCT = N./sqrt(X/df);
end
```

5.2 Density Approximation to the Noncentral t Distribution

```
% Calculates the log of the pdf for a NCT distribution
function pdfln = stdnctpdfln_j(x, nu, gam)
vn2 = (nu + 1) / 2; rho = x .^2;
pdfln = gammaln( vn2 ) - 1/2*log( pi *nu) - gammaln( nu / 2 ) - ...
    vn2*log1p ( rho / nu) ;
if ( all (gam == 0) ) , return , end
idx = ( pdfln ≥ -37) ; %   36.841   = log (1 e 16 )
if (any( idx ) )
     gcg = gam.^ 2 ; pdfln = pdfln - 0.5*gcg ; xcg = x .* gam;
     term = 0.5*log (2) + log (xcg) - 0.5*log (max( realmin , ...
         nu+rho ) ) ;
     term ( term == -inf ) = log ( realmin ) ; term( term == + ...
         inf ) = log ( realmax ) ;
     maxiter = 1e4 ; k = 0;
     logterms = gammaln ( ( nu+1+k ) / 2 ) - gammaln( k+1) - ...
         gammaln( vn2 ) + k*term ;
     fractions = real (exp( logterms ) ) ; logsumk = log ( ...
         fractions ) ;
     while ( k < maxiter)
          k = k + 1;
          logterms = gammaln ( ( nu+1+k ) / 2 ) - gammaln( k+1) - ...
              gammaln( vn2 ) + k*term( idx ) ;
          fractions = real (exp( logterms-logsumk( idx ) ) ) ;
          logsumk( idx ) = logsumk( idx ) + log1p ( fractions ) ;
          idx(idx) = (abs(fractions) > 1e-4) ; if ( all ( idx == ...
              false ) ) , break , end
     end
     pdfln = real ( pdfln+logsumk) ;
end
```

5.3 MLE for Parameters of Regular Location-Scale Student t Distribution

```matlab
% Computes MLE by optimising negative log-likelihood (nll) for ...
    location-scale t distribution
function MLE = tlikmax0(x, initvec)
tol =1e-5;
opts=optimset ( 'Disp ' , 'none ' , 'LargeScale ' , 'Off ' , ...
'TolFun ' ,tol , 'TolX ' ,tol , 'Maxiter ' ,200) ;
MLE = fminunc(@(param) tloglik(param ,x), initvec, opts);

function nll = tloglik(param, x)
v=param(1); mu=param(2); c=param(3);
if v<0.01, v=rand; end % An ad hoc way of preventing negative values
if c<0.01, c=rand; end % which works , but is NOT recommended !
K=beta (v/2 ,0.5) * sqrt( v ); z=(x-mu) / c;
ll = -log(c) -log (K) -(( v+1) /2) * log (1 + (z.^2) / v ); nll ...
    = -sum( ll );
% formula can be found in the statistical inference book on ...
    page: 430
% (last formula on page)
```

5.4 MLE for Parameters of Location-Scale Noncentral t Distribution

```matlab
% Computes MLE by optimising negative log-likelihood (nll) for ...
    location-scale nct distribution using the pdf provided in ...
    the book.
function MLE = nctlikmax_book(x, initvec)
tol =1e-5;
opts=optimset ( 'Disp ' , 'none ' , 'LargeScale ' , 'Off ' , ...
'TolFun ' ,tol , 'TolX ' ,tol , 'Maxiter ' ,200) ;
MLE = fminunc(@(param) nctloglik(param ,x), initvec, opts);

function nll = nctloglik(param, x)
df=param(1); mu=param(2); loc=param(3); scale=param(4);
if df<0.01, df=rand; end % An ad hoc way of preventing negative ...
    values
if scale<0.01, scale=rand; end % which works , but i s NOT ...
    recommended !
% this is simply the log applied to the pdf on page 373 in the ...
    intermediate
% probability book
z=(x-loc)/scale;
nll = -sum(-log(scale)+ stdnctpdfln_j(z,df,mu));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Using the function provided by Matlab
% Computes MLE by optimising negative log-likelihood (nll) for ...
    location-scale nct distribution using the pdf built-into Matlab.
function MLE = nctlikmax_matlab(x, initvec)
```

```
tol =1e-5;
opts=optimset ( 'Disp ' , 'none ' , 'LargeScale ' , 'Off ' , ...
'TolFun ' ,tol , 'TolX ' ,tol , 'Maxiter ' ,200) ;
MLE = fminunc(@(param) nctloglik(param ,x), initvec, opts);

function nll = nctloglik(param, x)
df=param(1); mu=param(2); loc=param(3); scale=param(4);
if df<0.01, df=rand; end % An ad hoc way of preventing negative ...
    values
if scale<0.01, scale=rand; end % which works , but i s NOT ...
    recommended !
% this is simply the log applied to the pdf on page 373 in the ...
    intermediate
% probability book
z=(x-loc)/scale;
nll = -sum(-log(scale)+log(nctpdf(z,df,mu)));
```

5.5 Sophisticated MLE for Parameters of Location-Scale Student t Distribution

```
% Computes more sophisticated MLE by optimising negative ...
    log-likelihood (nll) for location-scale nct distribution ...
    using the pdf provided in the book. It restricts the ...
    parameters in a smarter way and delivers approximate ...
    standard errors of the estimated parameters.
function [ param , stderr , iters , loglik , Varcov ] = ...
    tlikmax(x, initvec)

%%%%%%%%     df mu c
bound.lo=    [1 1 0.01];
bound.hi=    [100 1 100 ];
bound.which= [1 0 1 ];
% In this case , as bound . which for mu is zero , mu will not be
% restricted . As such , the values for .lo and .hi are irrelevant

maxiter =100; tol =1e-3; % change these as you see fit
opts=optimset('Display', 'notify-detailed', 'Maxiter', maxiter, ...
                 'TolFun', tol ,'TolX', tol, 'LargeScale', 'Off') ;
 [pout,fval,exitflag,theoutput,grad,hess]= ...
 fminunc(@(param) tloglik(param, x, bound), einschrk(initvec ...
     ,bound) ,opts) ;
V=inv(hess) ; % Don ' t negate : we work with the neg of the loglik
 [param, V]= einschrk (pout, bound, V) ; % Transform back , ...
     apply ∆ method
param=param' ; Varcov=V;
stderr=sqrt(diag(V)) ; % Approx std err of the params
loglik=-fval ; % The value of the loglik at its maximum .
iters=theoutput.iterations ; % Number of loglikfunction evals

function ll = tloglik(param , x , bound )
if nargin<3 , bound=0; end
if isstruct (bound) , paramvec=einschrk(real(param),bound,999);
else paramvec=param ;
end

```

```
v=paramvec( 1 ) ; mu=paramvec(2) ; c=paramvec(3) ;
K=beta (v /2, 0.5) * sqrt(v) ; z=(x-mu)/c ;
ll= -sum(-log(c)-log (K) - (( v+1) /2) * log (1 + (z.^2) / v ));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% function 'einschrk' is used to perform the conversion between ...
    theta and phi.
function [pout, Vout]= einschrk (pin, bound, Vin)
lo=bound.lo ; hi=bound.hi ; welche=bound.which;
if nargin < 3
    trans=sqrt((hi-pin)./(pin-lo) ) ; pout=(1-welche).*pin + ...
        welche.*trans ;
    Vout =[];
else
    trans=(hi+lo.*pin.^2) ./ (1+ pin .^2) ; pout=(1-welche).* ...
        pin + welche .* trans ;
    % now adjust the standard errors
    trans=2*pin .* (lo-hi ) ./ (1+pin .^2) .^2;
    d=(1-welche) + welche .* trans ; % either unity or ∆ method .
    J=diag (d) ; Vout = J* Vin * J ;
end
```

5.6 Sophisticated MLE for Parameters of Location-Scale Non-central t Distribution

```
% Computes more sophisticated MLE by optimising negative ...
    log-likelihood (nll) for location-scale nct distribution ...
    using the pdf provided in the book. It restricts the ...
    parameters in a smarter way and delivers approximate ...
    standard errors of the estimated parameters.
function [ param , stderr , iters , loglik , Varcov ] = ...
    tlikmax(x, initvec)

%%%%%%%%     df mu c
bound.lo=    [1 1 0.01];
bound.hi=    [100 1 100 ];
bound.which= [1 0 1 ];
% In this case , as bound . which for mu is zero , mu will not be
% restricted . As such , the values for .lo and .hi are irrelevant

maxiter =100; tol =1e-3; % change these as you see fit
opts=optimset('Display', 'notify-detailed', 'Maxiter', maxiter, ...
                'TolFun', tol ,'TolX', tol, 'LargeScale', 'Off') ;
 [pout,fval,exitflag,theoutput,grad,hess]= ...
 fminunc(@(param) tloglik(param, x, bound), einschrk(initvec ...
     ,bound) ,opts) ;
V=inv(hess) ; % Don ' t negate : we work with the neg of the loglik
 [param, V]= einschrk (pout, bound, V) ; % Transform back , ...
     apply ∆ method
param=param' ; Varcov=V;
stderr=sqrt(diag(V)) ; % Approx std err of the params
loglik=-fval ; % The value of the loglik at its maximum .
iters=theoutput.iterations ; % Number of loglikfunction evals

```

```
function ll = tloglik(param , x , bound )
if nargin<3 , bound=0; end
if isstruct (bound) , paramvec=einschrk(real(param),bound,999);
else paramvec=param ;
end

df=param(1); mu=param(2); loc=param(3); scale=param(4);
if df<0.01, df=rand; end % An ad hoc way of preventing negative ...
    values
if scale<0.01, scale=rand; end % which works , but i s NOT ...
    recommended !
% this is simply the log applied to the pdf on page 373 in the ...
    intermediate
% probability book
z=(x-loc)/scale;
% We are here using the density approximation provided in the book
nll = -sum(-log(scale)+ stdnctpdfln_j(z,df,mu));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% function 'einschrk' is used to perform the conversion between ...
    theta and phi.
function [pout, Vout]= einschrk (pin, bound, Vin)
lo=bound.lo ; hi=bound.hi ; welche=bound.which;
if nargin < 3
    trans=sqrt((hi-pin)./(pin-lo) ) ; pout=(1-welche).*pin + ...
        welche.*trans ;
    Vout =[];
else
    trans=(hi+lo.*pin.^2) ./ (1+ pin .^2) ; pout=(1-welche).* ...
        pin + welche .* trans ;
    % now adjust the standard errors
    trans=2*pin .* (lo-hi ) ./ (1+pin .^2) .^2;
    d=(1-welche) + welche .* trans ; % either unity or ∆ method .
    J=diag (d) ; Vout = J* Vin * J ;
end
```