NTNU
Norwegian University of
Science and Technology

Comparing the ACER and POT MCMC Extreme Value Statistics Methods Through Analysis of Commodities Data

Kristoffer Kofoed Rødvei

Master of Science in Physics and Mathematics
Submission date: April 2016
Supervisor: Arvid Næss, MATH
Co-supervisor: Andrea Riebler, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Comparing the ACER and POT MCMC Extreme Value Statistics Methods Through Analysis of Commodities Data

Kristoffer Kofoed Rødvei

April 2016

MASTER THESIS

Department of Mathematical Sciences

Norwegian University of Science and Technology

Supervisor: Arvid Næss

Secondary Advisor: Andrea Riebler

Summary and Conclusions

The Average Conditional Exceedance Rate and Peak Over Threshold Markov Chain Monte Carlo are two extreme value statistical methods, compared in this work. They are tested for both extrapolations and prediction intervals. The methods are compared for difference scenarios concluding that the Peak Over Threshold Markov Chain Monte generally prefered better for prediction intervals. It also seems to be preferable for extrapolation of independent and identically distributed data, and data approximately so. There are some indications that the Average Conditional Exceedance Rate method maybe favorable for capturing the data dependencies and extrapolation for correlated observations, but more work is needed for a conclusive result on that aspect.

Contents

Chapter 1

Introduction

Extreme value statistics is the part of statistics dealing with extremely large or small events, which deviates heavily from the distribution median. By the asymptotic theory, if there exists a limiting distribution, the distribution converge towards the Gumbel, Fréchet or Weibull, or the combined Generalized Extreme Value (GEV) distribution at the upper and lower limits (Coles, 2001, p. 46). For parameter estimation the observations are required to be independent and identically distributed (i.i.d). The observed data are often taken close in time, which normally violate the assumption of independence. Hence filtrating methods like the block maxima and r largest order statistics (Coles, 2001, p. 66) are commonly applied in an effort to remove dependency. Unfortunately, both filtering methods are quite wasteful, resulting in only a small number of observation being used for parameter estimation. The Peak Over Threshold (POT) is an alternative statistical method developed from the GEV distribution. The POT method only uses data above a given threshold, which normally makes the filtering less wasteful resulting in an increased estimation accuracy. For the possibilities of implying prior beliefs and physical knowledge, the Bayesian statistics calculated using the Adaptive Markov Chain Monte Carlo (AMCMC) method is developed for the POT. The Bayesian approach also gives an opportunity for more freely inference for special cases, as PI estimation. The POT MCMC method is compared to another alternative method called the Average Conditional Exceedance Rate (ACER) introduced by Næss and Gaidai (2009). This method does not rely on the assumption of independence. Hench enabling the possibilities of capturing the data dependency without wasteful filtering. To a certain extent the ACER method also have the capability of capturing the subasymptotic parts.

The development of the Bayesian statistics and MCMC method for the GEV distribution

was presented in (Coles, 2001, p. 170), while no information was found for the POT AMCMC method. The ACER method was applied for multiple scenarios in the paper by Næss et al. (2013), and for wind speed data in Karpa and Næss (2012). Raw data and AR-GARCH filtrated data was evaluated at the VaR tails for the ACER, POT, normal and t-distribution by Dahlen et al. (2015) and Dahlen (2010), with a resulting conclusion that both the ACER and POT method outperform the normal and t-distribution at the tails. There were signs that the ACER method overall performed better than the POT method, but not significantly better. The papers by Giot and Laurent (2003), Aloui and Mabrouk (2010) and Steen et al. (2015), presented methods for analyzing the VaR of commodity data at the tails. Giot and Laurent (2003) concluded that the AP-APARCH skewed student t-distributed filtration for commodity data was superior, compared to a number of different methods for heteroscedasticity filtering.

As no comparison between the ACER and POT MCMC method, to date have been found, the results can be beneficial for model selection of future analysis. The goal is to make sufficient statistical evidence to suggest one of the two methods for different scenarios. The POT MCMC and ACER methods are compared through the Value at Risk (VaR) extrapolation and Prediction Interval (PI) of future extremes through different scenarios. The method for calculating PI of future extreme was developed in this work, as little information was found on the subject. The POT MCMC and ACER method was tested on i.i.d synthetic Pareto distributed data sets, dependent synthetic Autoregressive(AR) Asymmetric Power Autoregressive Conditional Heteroscedasticity (APARCH) data, daily return of commodities and AR-APARCH filtered daily return of commodities.

Chapter 2

Theory

The following chapter gives an introduction to the theory behind this work. Some of the sections only gives a brief description of the theory, for additional details the reader is referred to the suggested litterature.

2.1 Extreme value theory

The essence of extreme value theory is to analyze the maximum of a series of random variables $X_1, \ldots, X_n$. Defining M_n as the maximum of a sequence

$$M_n = \max\{X_1, \ldots, X_n\},$$

the resulting distribution of M_n is

$$\Pr(M_n \leq z) = \Pr(X_1 \leq z, \ldots, X_n \leq z). \tag{2.1}$$

By assuming $X_1, \ldots, X_n$ independent and identically distributed (i.i.d.) with common cumulative distribution function F, equation (2.1) reduces to

$$\begin{aligned} \Pr(M_n \leq z) &= \Pr(X_1 \leq z) \times \cdots \times \Pr(X_n \leq z) \\ &= \big[F(z)\big]^n. \end{aligned} \tag{2.2}$$

The distribution F is normally unknown, and a small error in the estimated distribution F can escalate to a large error in the resulting F^n.

As for the central limit theory for the normal distribution, extreme value theory also has

a limiting distribution by the Fisher–Tippett–Gnedenko theorem. If there exist values $a_n > 0$ and b_n such that the

$$\lim_{n\to\infty} \Pr\left[(M_n - b_n)/a_n \le z\right] \to G(z),$$

where G is a non-degenerating distribution function, then G is on one of the forms

$$G(z) = \exp\left\{-\exp\left[-\left(\frac{z-b}{a}\right)\right]\right\}, \qquad -\infty < z < \infty; \tag{2.3}$$

$$G(z) = \begin{cases} 0, & z \le b, \\ \exp\left\{-\left(\frac{z-b}{a}\right)^{-\alpha}\right\}, & z > b; \end{cases} \tag{2.4}$$

$$G(z) = \begin{cases} \exp\left\{-\left[-\left(\frac{z-b}{a}\right)\right]^{\alpha}\right\}, & z \le b, \\ 1, & z > b; \end{cases} \tag{2.5}$$

for parameters $a > 0$, b and $\alpha > 0$. Here equation (2.3), (2.4) and (2.5) refers to Gumbel, Fréchet and Weibull distribution respectively. The above equations can be combined into the General Extreme Value (GEV) distribution

$$G(z) = \exp\left\{-\left[1 + \xi\left(\frac{z-\mu}{\sigma}\right)\right]^{-\frac{1}{\xi}}\right\} \tag{2.6}$$

where the location parameter is $-\infty < \mu < \infty$, the scale parameter is $\sigma > 0$ and the shape parameter is $-\infty < \xi < \infty$. For the expression to be valid, the requirement $1 + \xi(z-\mu)/\sigma > 0$ must be fulfilled. It can easily be verified that the GEV equals equation (2.4) when $\xi > 0$, equation(2.5) when $\xi < 0$ and converges towards equation(2.3) as $z \to 0$.

For parameter estimation the observation is required to be i.i.d., as it was assumed for the development of equation 2.2. Unfortunately, in practice observations are commonly dependent. Block maxima and r largest order statistics are common methods for filtering the dependent data points into an approximate i.i.d. dataset (Coles, 2001, p. 66). The basic principle is to only use the largest, or r largest data within each block. Examples of block sizes could be week, month, year etc. For a deeper description of extreme value theory, GEV or block maxima, the book of (Coles, 2001, Chapter 3) is suggested.

2.1.1 Peak Over Threshold

One of the problems with the GEV method and the i.i.d. filtration of data points, is that the block maxima and r largest order statistics are quite wasteful. Especially in situations where some blocks contain a larger number of extremes than others. Large extremes will be discarded from the set, which otherwise would have been accepted in other blocks.

The Peak Over Threshold (POT) method is suggesting a different method of tackling the i.i.d. filtration. The POT method filters the data by only using points over a certain threshold u, avoiding the problem of discarding large extremes in certain blocks. As long as u is chosen sufficiently large, the resulting data will be i.i.d. It can be shown by the GEV distribution equation (2.6), like was done by (Coles, 2001, p. 76), that

$$\Pr(X > y + u | X > u) = \left(1 + \frac{\xi y}{\tilde{\sigma}}\right)^{-\frac{1}{\xi}}, \tag{2.7}$$

where y is the threshold excess, given by $y = z - u$ for $z > u$. The resulting cumulative distribution of $X - u$ is called the Generalized Pareto Distribution (GPD)

$$H(y) = \begin{cases} 1 - \left(1 + \frac{\xi y}{\tilde{\sigma}}\right)^{-\frac{1}{\xi}} & \text{if } \xi \neq 0, \\ 1 - \exp\left(-\frac{y}{\tilde{\sigma}}\right) & \text{if } \xi = 0, \end{cases} \tag{2.8}$$

where ξ equals the GEV parameter, while $\tilde{\sigma} = \sigma + \xi(u - \mu)$. The conditional probability is reduced since $\Pr(X > y + u, X > u) = \Pr(X > y + u)$

$$\begin{aligned} \Pr(X > y + u | X > u) &= \frac{\Pr(X > y + u, X > u)}{\Pr(X > u)} \\ &= \frac{\Pr(X > y + u)}{\Pr(X > u)}. \end{aligned} \tag{2.9}$$

By combining equation (2.7) and (2.9), the probability of a future event can be found by

$$\begin{aligned} \Pr(X > z | \xi, \sigma) &= \Pr(X > y + u) \\ &= \Pr(X > u) \cdot P(X > y + u | X > u) \\ &= \Pr(X > u) \cdot \left[1 + \xi\left(\frac{z - u}{\tilde{\sigma}}\right)\right]^{-\frac{1}{\xi}}, \end{aligned} \tag{2.10}$$

where $\Pr(X > u)$ is the probability that a random point exceeds the threshold. For more

information about the POT method, see (Coles, 2001, Chapter 4).

Declustering

The number of threshold excesses y increases as the threshold u decreases. A larger number of threshold excesses will increase the accuracy and lower the variance of the parameter estimation, which suggests using a low threshold. In practice, data are often correlated, heteroscedastic or nonstationary. For data without trend, a high enough threshold will ensure close to i.i.d. property for the threshold excess. As the threshold decreases, clusters could appear, and the threshold excesses will no longer be i.i.d. Violation of the i.i.d. property will result in an estimation bias, which suggests using a high threshold. The selection of threshold comes down to the trade-off between accuracy and bias. The goal is to get the lowest variance without bias.

A declustering method can be applied to improve the i.i.d. property for low threshold. The target is to localize clusters above the threshold and select the largest value within each cluster. The method used here defines a cluster as the points above the threshold until r consecutive points are observed below. Referring to figure 2.1 as an example on how the method is used in practice. For $r = 2$ there are 7 clusters, while for $r = 4$ there are 3. For the particular threshold used in the plot, $r = 4$ seems like the obvious choice. For the POT where declustering is used, the only observation used for parameter estimation is cluster maximum, hence the condition in equation (2.9), will become X is a cluster maximum. The change will follow for the condition in equation (2.10) as well as $\Pr(X > u)$. For a more in depth description of declustering, see (Coles, 2001, p. 100). The declustering `R` code can be found in appendix B.

Threshold

As stated above, the goal when selecting threshold u is to find the smallest threshold u_0, for which the model is still unbiased, such that the highest accuracy is achieved. For this paper, the combination of two methods for threshold selection are used.

The first method uses the fact that the mean of the GPD

$$E(Y) = \frac{\tilde{\sigma}}{1-\xi}, \tag{2.11}$$

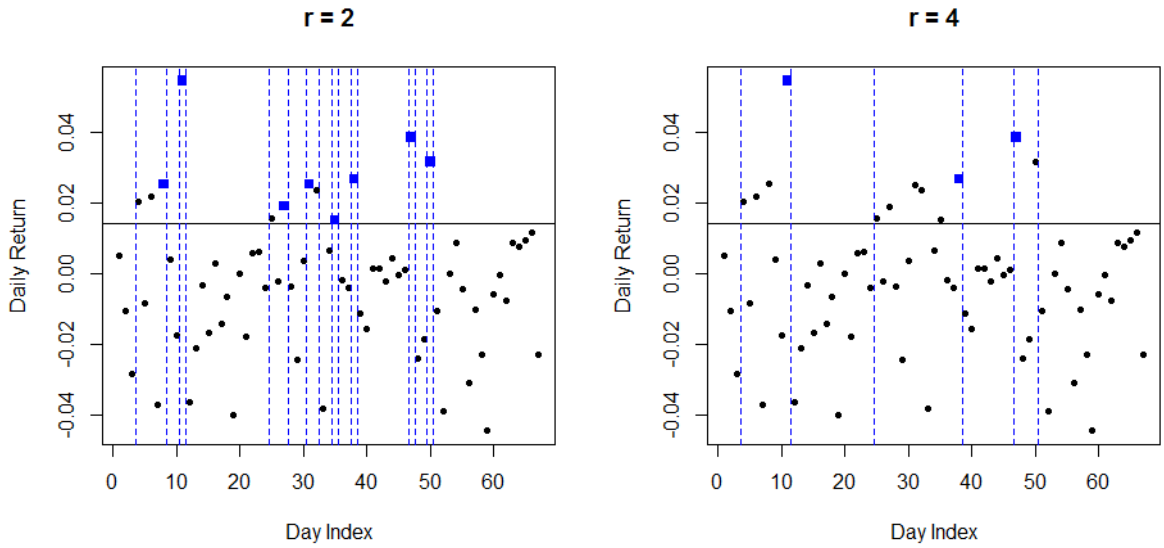

Figure 2.1: Portion of the crude oil daily return series, described in chapter 3.2. The horizontal solid line is threshold, with $u = 0.014$. Clusters are localized between the vertical blue dashed lines, with the largest value within each clusters shown as a blue square.

for $\xi < 1$, and infinite when $\xi > 1$. Thus the first method fails when $\xi > 1$, but in practice ξ rarely exceeds 1. As shown above, the GPD ξ equals the GEV parameter which is independent of threshold, while $\tilde{\sigma} = \sigma + \xi(u - \mu)$ is linear with respect to threshold. Here σ and μ are the GEV parameters and are independent of the threshold. Thereby the mean of Y is also linear proportional to the threshold. By plotting the mean of threshold excess against thresholds, linear effect should be apparent from u_0. A confidence interval can be added for a better understanding of where the linearity starts. For larger values of thresholds, there will only be a few number of threshold excesses, hence it is suggested using the t-distribution with 1 degree of freedom for the $E(Y)$ confidence intervals estimation. An example of the mean excess plot can be seen in figure 4.8. For more information about the method, see (Coles, 2001, p. 79). The `R` code for mean excess plot, can be found in appendix B.

For the second method, estimates of ξ and $\tilde{\sigma}$ are taken for a variety of thresholds. The parameters ξ and the reparametrized $\sigma^* = \tilde{\sigma} - \xi u$ should both be constant from u_0. For pinpointing u_0, both ξ and σ^* is plotted against u, with added confidence intervals. An example plot of the shape and modified scale parameter against threshold can be seen in figure 4.9. For more information about the method, see (Coles, 2001, p. 83). The `R` code for the parameter plot can be found in appendix B.

After threshold selection, $\tilde{\sigma}$ is simply estimated from the threshold excess and is required

larger than zero. For simplicity, from here, the notation σ is used for the GPD parameter $\tilde{\sigma}$, as long as otherwise is not specified.

2.1.2 Average Condition Exceedance Rate

The Average Conditional Exceedance Rate (ACER) is another extreme value method, first introduced by Næss and Gaidai (2009). For a more in depth description of the ACER method see Næss et al. (2013). Both the GEV and GPD distributions require the observations to be i.i.d. When observations are not i.i.d., filtering methods such as threshold exceedance, declustering, blocking etc., are used to achieve close to i.i.d. data. The problem with these filtering methods is that they often discard most of the data, such that only a small amount of the data can be used for parameter estimation. The advantage of the ACER method is that the observations are not restricted to i.i.d. or even stationarity data as long as the data has no trend. Another advantage is the ACER method's ability to a certain extent capture the subasymptotic parts, which also can improve estimation.

Without the i.i.d. assumption for $X_1,\dots,X_n$, equation (2.1) can be written using time dependency

$$\Pr(M_n \le z) = \prod_{j=2}^{n} \Pr(X_j \le z, | X_{j-1} \le z, \dots, X_1 \le z) \cdot \Pr(X_1 \le z). \tag{2.12}$$

It is reasonable to assume that the data dependency with neighboring points decrease by time, and is negligible after $k \ll n$ steps, such that
$\Pr(X_j \le z, | X_{j-1} \le z, \dots, X_1 \le z) \approx \Pr(X_j \le z, | X_{j-1} \le z, \dots, X_{j-k+1} \le z)$, for every $j = k, \dots, n$. Using this and Taylor expansion of the exponential function around zero, equation (2.12) reduces to

$$\begin{aligned} \Pr(M_n \le z) &\approx \exp\left(- \sum_{j=k}^{n} \alpha_{kj}(z) - \sum_{i=1}^{k-1} \alpha_{ii}(z) \right) \\ &\approx \exp\left(- \sum_{j=k}^{n} \alpha_{kj}(z) \right) \end{aligned} \tag{2.13}$$

where $\alpha_{kj}(z) = \Pr(X_j \ge z | X_{j-1} \le z, \dots, X_{j-k+1} \le z)$ for $k \ge 2$ and $\alpha_{kj}(z) = \Pr(X_j \ge z)$ for $k = 1$. The final step is justified since $\sum_{i=1}^{k-1} \alpha_{ii}(z)$ is negligible compared to $\sum_{j=k}^{n} \alpha_{kj}(z)$ for $k \ll n$, while the Taylor expansion around zero is reasonable at the upper tail since for large z, $\alpha_{kj}(z)$ is close to zero.

Considering the ACER as

$$\epsilon_k(z) = \frac{1}{n-k+1} \sum_{j=k}^{n} \alpha_{kj}(z). \tag{2.14}$$

The ACER function can be estimated using

$$\hat{\epsilon}_k(z) = \frac{\sum_{j=k}^{n} \mathbf{1}(x_j \geq z, x_{j-1} \leq z, \ldots, x_{j-k+1} \leq z)}{\sum_{j=k}^{n} \mathbf{1}(x_{j-1} \leq z, \ldots, x_{j-k+1} \leq z)}. \tag{2.15}$$

where $\mathbf{1}(\omega)$ is the indicator function for event ω. For nonstationary observations it is suggested using $n-k+1$ as an approximation for the denominator. The approximation can be justified since $\mathbf{1}(x_{j-1} \leq z, \ldots, x_{j-k+1} \leq z) \to 1$ in the upper tail where z is large.

It is assumed that the tail of the ACER function follows

$$\epsilon_k(z|a_k, b_k, c_k, q_k, \xi_k) = q_k \left[1 + \xi_k \left(a_k (z - b_k)^{c_k}\right)\right]^{-1/\xi_k}, \tag{2.16}$$

where the parameters a_k, b_k, c_k, q_k and ξ_k are approximately constant in the upper tail for a certain k. The process of selecting k can be done by investigating the plot of $\hat{\epsilon}_k(z)$ against z for a variety of k, k is set to the smallest value for which increasing k makes negligible change to the tail. A k-plot example is shown in figure 4.5. The parameters can be estimated by minimizing the weighted square error

$$F(a, b, c, q, \xi) = \sum_{i=1}^{N} w_i \left[\log(\hat{\epsilon}_k(z_i)) - log(q) + \xi^{-1} \log\left(1 + a(z_i - b)^c\right)\right]^2, \tag{2.17}$$

using numerical methods. Selecting $z_1, \ldots, z_N$ is done by uniformly dividing the values from where regular tail behavior of $\hat{\epsilon}_k(z)$ starts z_1 to $\max_{1 \leq i \leq n}(X_i)$ into N points. The weights w_i is calculated using

$$w_i = \left(\log\left[C_\alpha^+(z_i)\right] - \log\left[C_\alpha^-(z_i)\right]\right)^{-2}, \tag{2.18}$$

where $C_\alpha^+(z_i)$ and $C_\alpha^-(z_i)$ is the upper and lower $100 \cdot \alpha\%$ confidence interval values respectively for $\hat{\epsilon}_k(z_i)$. Organizing the observation into R similar realizations, like R years, the sample variance can be calculated as

$$\hat{s}_k(z_i)^2 = \frac{1}{R-1} \sum_{r=1}^{R} \left(\hat{\epsilon}_k^{(r)}(z_i) - \hat{\epsilon}_k(z_i)\right) \tag{2.19}$$

where $\hat{\epsilon}_k^{(r)}(z_i)$ is the estimated ACER function for the r realization at z_i. Hence a $100 \cdot \alpha\%$ confidence interval can be calculated using the student t-distribution

$$C_\alpha^{\pm}(z_i) = \hat{\epsilon}_k(z_i) \pm t_{(1-\alpha)/2,R-1} \frac{\hat{s}_k(z_i)}{\sqrt{R}} \tag{2.20}$$

where $t_{p,v}$ is defined as $\Pr(T > t_{p,v}) = p$ for the standardized t-distribution with v degrees of freedom.

After parameter estimation a future prediction can be achieved using equation (2.16). Confidence intervals can be added to the ACER function prediction by estimating the parameters to the upper and lower confidence curve. Using $\epsilon_k(z_i|a,b,c,q,\xi) \pm t_{(1-\alpha)/2,R-1} \frac{\hat{s}_k(z_i)}{\sqrt{R}}$ instead of $\hat{\epsilon}_k(z_i)$ in equation (2.17), where $\epsilon_k(z_i|a,b,c,q,\xi)$ is given by equation (2.16), parameters for upper and lower confidence curves are estimated. These upper and lower confidence parameters can be used in equation (2.16) for ACER function out of sample confidence intervals. The majority of ACER calculation was achieved using the ACER `R` package by Rødvei (2015).

2.2 Bayesian Inference

For the traditional frequentist statistics, the parameters $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_m]$ are assumed fixed, while observations $\boldsymbol{x} = [x_1, \ldots, x_n]$ are random from the underlying distribution $f(\boldsymbol{x}|\boldsymbol{\theta})$. Bayesian statistics instead treats the parameters $\boldsymbol{\theta}$ with a probability distribution, where it is possible to make subjective beliefs about the distribution, independent of the data. These subjective beliefs are used to construct a prior distribution $f(\boldsymbol{\theta})$ based on experience, information or physical knowledge of the situation analyzed.

The posterior distribution of the parameters, dependent on the observed data becomes

$$f(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{f(\boldsymbol{\theta}) f(\boldsymbol{x}|\boldsymbol{\theta})}{\int_\Theta f(\boldsymbol{\theta}) f(\boldsymbol{x}|\boldsymbol{\theta}) \mathrm{d}\theta}, \tag{2.21}$$

where Θ is the domain over all possible parameters for which the integral is taken, and $f(\boldsymbol{x}|\boldsymbol{\theta})$ is the likelihood function. The likelihood function is constructed from the joint density function, which for independent data equals

$$f(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{n} f(x_i|\boldsymbol{\theta}). \tag{2.22}$$

The integral over parameters reduces to a constant, which makes

$$f(\boldsymbol{\theta}|\boldsymbol{x}) \sim c \cdot f(\boldsymbol{\theta}) f(\boldsymbol{x}|\boldsymbol{\theta}), \tag{2.23}$$

where $c = 1/\int_\Theta f(\boldsymbol{\theta}) f(\boldsymbol{x}|\boldsymbol{\theta}) \mathrm{d}\theta$ is the normalizing constant.

A conjugate prior is a prior which combined with the likelihood function constructs a posterior distribution in the same family as the prior. Conjugate priors are often preferred because of the analytical luxury and computational simplicity.

Estimating a future point z which follows the distribution $f(z|\boldsymbol{\theta})$, can then be done using the posterior distribution $f(\boldsymbol{\theta}|\boldsymbol{x})$. The predicted point then becomes dependent of the observed data

$$f(z|\boldsymbol{x}) = \int_\Theta f(z|\boldsymbol{\theta}) f(\boldsymbol{\theta}|\boldsymbol{x}) \mathrm{d}\theta. \tag{2.24}$$

Since Bayesian inference accounts for the distribution of parameters, equation (2.10) can be rewritten using $\Pr(Z > u) = \psi$

$$\Pr(Z > z|\xi, \sigma, \psi) = \psi \left[1 + \xi \left(\frac{z-u}{\sigma}\right)\right]^{-\frac{1}{\xi}}, \tag{2.25}$$

where ψ, ξ and σ are the unknown parameters. Since ψ is the probability of a point being larger than the threshold, ψ is independent of ξ and σ. Development of posterior distributions for independent parameters can be treated separately.

Starting with ξ and σ, by combining equation (2.22) and (2.8), the joint density function for the POT method becomes

$$\begin{aligned} f(\boldsymbol{y}|\xi, \sigma) &= \prod_{i=1}^{n} h(y_i|\xi, \sigma) \\ &= \sigma^{-n} \prod_{i=1}^{n} \left(1 + \frac{\xi y_i}{\sigma}\right)^{-\left(1+\frac{1}{\xi}\right)}, \end{aligned} \tag{2.26}$$

or $f(\boldsymbol{y}|\sigma) = \sigma^{-n} \exp\left\{-\sigma^{-1} \sum_{i=1}^{n} y_i\right\}$ when $\xi = 0$. Here h is the probability density function of the GPD, $\boldsymbol{y}$ is a vector of observed threshold excess and n is the numbers of threshold excess.

The obvious start for investigating priors is the conjugate priors. but unfortunately, there do not appear to be any conjugate priors for the joint GPD. This paper, will not go into depth

on how to select Bayesian priors for the GPD, but instead use the suggestion proposed by (Coles, 2001, p. 174). Note that there are potential improvements by deeper investigation of GPD or GEV priors, especially for priors developed for specific situations where there are physical knowledge or practical experiences about the parameters. Since $\sigma > 0$, the transformation $\phi = \log(\sigma)$ ensures σ to be valid without restriction on ϕ. The suggested priors are $f_\phi(\cdot)$ and $f_\xi(\cdot)$ to be normally distributed around zero with variance $v_\phi = 10^4$ and $v_\xi = 100$.

Considering the prior distribution of ϕ instead of σ, the change of variable for the joint density function becomes

$$\begin{aligned} f_{\boldsymbol{y}|\xi,\phi}(\boldsymbol{y}|\xi,\phi) &= \frac{f_{\boldsymbol{y},\xi,\phi}(\boldsymbol{y},\xi,\phi)}{f_{\xi,\phi}(\xi,\phi)} \\ &= \frac{f_{\boldsymbol{y},\xi,\sigma}\left(\boldsymbol{y},\xi,\exp(\phi)\right)\cdot\left|\frac{\mathrm{d}}{\mathrm{d}\phi}\exp(\phi)\right|}{f_{\xi,\sigma}\left(\xi,\exp(\phi)\right)\cdot\left|\frac{\mathrm{d}}{\mathrm{d}\phi}\exp(\phi)\right|} \\ &= f_{\boldsymbol{y}|\xi,\sigma}\left(\boldsymbol{y}|\xi,\exp(\phi)\right), \end{aligned} \tag{2.27}$$

where $f_X(\cdot)$ indicates the probability distribution of X. The posterior distribution of the parameters can then be developed by the priors, (2.27) and (2.23)

$$f_{\xi,\phi|\boldsymbol{y}}(\xi,\phi|\boldsymbol{y}) \sim c\cdot f_{\boldsymbol{y}|\xi,\sigma}\left(\boldsymbol{y}|\xi,\exp(\phi)\right) f_\xi(\xi) f_\phi(\phi), \tag{2.28}$$

where again c is the normalizing constant, $f_{\boldsymbol{y}|\xi,\sigma}$ as in (2.26), $f_\xi(\xi) \sim N(0,100)$ and $f_\phi(\phi) \sim N(0,10^4)$. Here $N(\mu,\sigma^2)$ indicates the normal distribution with mean μ and variance σ^2.

The development of ψ posterior distribution can be started with investigating priors. Since ψ equals $\Pr(X > u)$, the range is limited within 0 and 1. For simplicity the prior is set proportional to the uniform distribution on the interval $(0,1)$, $f(\psi) \sim \mathrm{UNIF}(0,1)$. It is noted that in reality the distribution of ψ is not flat. Low valued ψ is more likely than high, while the probability converges to zero for the endpoints. A well-tuned Beta distributed prior could account for this and improve the result.

The joint density function can be created by the fact that ψ equals the probability that a random event exceeds the threshold. This can be expressed using the binominal distribution, where k_i indicates the numbers of points exceeding the threshold and N_i indicates the total numbers of points, each for a given period i. For a total m numbers of periods the

posterior distribution equals

$$\begin{aligned} f(\psi|k_1,\ldots,k_n,N_1,\ldots,N_n) &\sim f(\psi)\cdot\prod_{i=1}^{m} f(k_i|N_i,\psi) \\ &= \prod_{i=1}^{m}\binom{N_i}{k_i}\psi^{k_i}(1-\psi)^{N_i-k_i} \\ &\sim \psi^{\sum_{i=1}^{m}k_i}(1-\psi)^{\sum_{i=1}^{m}N_i-\sum_{i=1}^{m}k_i}. \end{aligned} \tag{2.29}$$

The resulting distribution is independent of period selection, the notation k and N can be used for the total numbers of exceedance and the total numbers of observations respectively. It is noted that the distribution is proportional to the Beta distribution. Since there only exist one normalizing constant which satisfies the requirements for a probability distribution, the posterior distribution is not only proportional, but equal to the Beta distribution. Rewriting equation (2.29) gives

$$f(\psi|k,N) \sim \text{BETA}(k+1, N-k+1). \tag{2.30}$$

For declustered data, the parameters ξ and σ is estimated as described above. Approaching the ψ as above, k and N equals the number of cluster maximum and total number of observations respectively.

2.3 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a powerful iterative method used to sample from a probability distributions, which analytically or through other simulation methods can be difficult and impractical to sample from. The algorithm is constructed by converging the target distribution to an irreducible and aperiodic Markov Chain with limiting distribution equal the target distribution. Independent of starting position the Markov Chain will then converge towards the desired probability distribution in the limit as the numbers of iterations goes to infinity. The first numbers of realizations of the Markov Chain until convergence is called burn-in. These numbers are discarded for further analyses. More information about burn-in can be found in (Givens and Hoeting, 2013, p.220). The remaining realizations approximately follow the target probability distribution, where the accuracy increases as the numbers of realizations increases. The Monte Carlo method can then be used to calculate the quantities of interest like mean, expected value, future prediction, credible interval, pre-

diction interval etc. More in-depth description of the MCMC method can be found in the books of Gamerman and Lopes (2006) and (Givens and Hoeting, 2013, Chapter 7,8).

2.3.1 Gibbs Sampling

In situations where it is difficult to sample from the joint distribution, but applicable from the conditional distribution, Gibbs sampling is preferable. The theory behind Gibbs sampling was first proposed in Geman and Geman (1984). The principle of Gibbs sampling is to construct the Markov Chain by repeatedly sample each parameter with the rest of the parameters as the condition. The Gibbs sampler starts with an initial guess $\boldsymbol{X}^0 = [X_1^0, \cdots, X_n^0]$, and is iteratively updated by the following scheme

$$\begin{aligned} X_1^{t+1}|\cdot &\sim f(X_1|X_2^t, \ldots, X_n^t), \\ X_2^{t+1}|\cdot &\sim f(X_2|X_1^{t+1}, X_3^t, \ldots, X_n^t), \\ &\vdots \\ X_n^{t+1}|\cdot &\sim f(X_n|X_1^{t+1}, \ldots, X_{n-1}^{t+1}), \end{aligned} \tag{2.31}$$

where t is the iteration number, f is probability function of the parameter and $|\cdot$ symbolizes that the function is conditional on the rest and recent parameters. In some cases it can be beneficial to sample some of the parameters in blocks, such as $(X_k, X_{k+1})|\cdot$ where $1 \leq k \leq n-1$. This form of Gibbs sampling is called blocking. The iterative process is repeated until enough realizations are generated for sufficient accuracy. More about Gibbs sampling can be found in (Gamerman and Lopes, 2006, p. 141) and (Givens and Hoeting, 2013, p. 209)

2.3.2 Metropolis–Hastings Algorithm

The Metropolis-Hastings algorithm was first proposed by Metropolis et al. (1953), and is another method for constructing a suitable Markov Chain. The algorithm is preferable for situations where a proportional distribution is simple to evaluate, while the target probability distribution is difficult. Bayesian inference see chapter 2.2, often results in a distribution where the normalizing constant cannot analytically be calculated. While possible numerically, the normalizing constant often becomes computationally expensive, which makes it impractical for iterative simulations. Using the Metropolis–Hastings algorithm on a proportional distribution without normalizing constant, results in samples from the target distri-

butions.

The Metropolis–Hastings algorithm starts with an initial guess for the parameters. For each iteration new parameters $\boldsymbol{X}^*$ are suggested from a proposal distribution $g(\boldsymbol{X}^*|\boldsymbol{X}^t)$, given the last accepted parameter $\boldsymbol{X}^t$. The new parameter is then evaluated against the last accepted by

$$R(\boldsymbol{X}^*, \boldsymbol{X}^t) = \frac{f(\boldsymbol{X}^*)g(\boldsymbol{X}^t|\boldsymbol{X}^*)}{f(\boldsymbol{X}^t)g(\boldsymbol{X}^*|\boldsymbol{X}^t)} \tag{2.32}$$

where $f(x)$ is the target distribution, or a distribution proportional to the target distribution. The parameter $\boldsymbol{X}^{t+1}$ takes value $\boldsymbol{X}^*$ with probability $\min\{1, R(\boldsymbol{X}^*, \boldsymbol{X}^t)\}$, if rejected we then get $\boldsymbol{X}^{t+1} = \boldsymbol{X}^t$ instead. The reason that the normalizing constant in the target distribution is irrelevant is because they are both canceled out in $f(\boldsymbol{X}^*)/f(\boldsymbol{X}^t)$.

A common proposal distribution is the random walk. The new parameters are generated from the last accepted realization with additional variance, $\boldsymbol{X}^* = \boldsymbol{X}^t + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ follows a chosen probability distribution. Symmetric proposals implies that $g(\boldsymbol{X}^t|\boldsymbol{X}^*) = g(\boldsymbol{X}^*|\boldsymbol{X}^t)$. This is referred to as Metropolis algorithm.

The Metropolis–Hastings algorithm can in situations be necessary for some of the steps in the Gibbs sampler, equation (2.31). Such a combination of Metropolis-Hastings algorithm and Gibbs sampler is referred to as a Hybrid Gibbs sampler, and was first introduced by Müller (1991). For more information about the Metropolis-Hastings algorithm and Hybrid Gibbs sampler see (Givens and Hoeting, 2013, p. 202), (Gamerman and Lopes, 2006, p. 191) and (Givens and Hoeting, 2013, p. 216), (Gamerman and Lopes, 2006, p. 205) respectively.

2.3.3 Effective Sample Size

The realizations of the simulated Markov Chain will often be correlated, and dependent on the future and past iterations. The correlation implies that the information gained by each iteration is less than the suggested run length. The effective sample size gives a method of calculating the theoretical size of an equally informative i.i.d. realization set. The effective sample size is estimated as

$$L_{\text{eff}} = \frac{L}{1 + 2\sum_{k=1}^{K} \hat{\rho}(k)}, \tag{2.33}$$

where L is the sample size of the simulated realizations, $\hat{\rho}(k)$ is the estimated k step autocorrelation between realizations and K is chosen as the first k where $\hat{\rho}(k) < 0.1$. The effective sample size is a quantification of the information held by the simulated realization set. The

R code for calculating the effective sample size, can be found in appendix B.

2.3.4 Adaptive Metropolis Algorithm

A challenge with constructing an MCMC is to ensure that the series converges to the stationary target distribution relatively quickly, and that the samples give points in the whole range of the target distribution. This is referred to as good mixing.

If a large percentage of the Metropolis-Hastings proposals $\boldsymbol{X}^*$ is accepted, the proposal distribution is too narrow. High acceptance rate will delay convergence, and cause higher correlation between points. The result is poor mixing and a decrease in effective sample size. On the other hand, if only a small percentage of the proposals are accepted, the proposal distribution is to wide. Low acceptance rate will also increase correlation, which gives poor mixing and decreased effective sample size. A large number of generated realizations by the Markov chain will be equal, which will harm future Monte Carlo simulation.

To maximize the effective sample size and ensure good mixing, the acceptance rate should be somewhere in between. For a Metropolis-Hastings algorithm, Gelman et al. (1996) suggested a 44% acceptance rate for single dimensional normal target distribution and 23.4% for high dimensional multivariate normal target distribution. Commonly the user would run the Metropolis-Hastings algorithm, calculate acceptance rate, tune variance and then rerun the process until sufficient acceptance rate is achieved.

For this work, MCMC simulation will be used for a large number of different situations, and it would become extremely time-consuming to tune each variance. This inconvenience can be handled by using an Adaptive Markov Chain Monte Carlo (AMCMC) which adapt the MCMC algorithm while running. This is achievable using a normal random walk proposal where the next suggested realization $\boldsymbol{X}^* \sim N(\boldsymbol{X}^t, \lambda\boldsymbol{\Sigma}^t)$. Between iterations $\boldsymbol{\Sigma}^t$ is adjusted to improve mixing and efficient sample size. The acceptance rate is set by λ, and with a p dimensional multivariate normal target distribution, it has been shown that a constant $\lambda = 2.38^2/p$ is optimal when $\boldsymbol{\Sigma}$ equals the real variance of the target distribution (Gelman et al., 1996). The adaptive Metropolis algorithm is not constrained to the normal target distributions, but the suggested λ seems like a good starting value. The ability of an adjustable λ between future iteration seems beneficial, because of the unknown target distribution and the following acceptance rate. The additional adaptive parameter $\boldsymbol{\mu}^t$ is necessary since the covariance is proportional to $\boldsymbol{\mu}$. The initial guess is chosen as $\boldsymbol{\mu}^0 = \mathbf{0}$ and $\boldsymbol{\Sigma}^0 = \mathbf{I}$. The nor-

mal random walk proposal distribution is symmetric, which result in an adaptive Metropolis algorithm, where (2.32) is reduced to

$$R(\boldsymbol{X}^*, \boldsymbol{X}^t) = \frac{f(\boldsymbol{X}^*)}{f(\boldsymbol{X}^t)}. \tag{2.34}$$

For each iteration $\boldsymbol{\mu}^{t+1}$ and $\boldsymbol{\Sigma}^{t+1}$ is updated as follows

$$\boldsymbol{\mu}^{t+1} = \boldsymbol{\mu}^t + \gamma^{t+1}(\boldsymbol{X}^{t+1} - \boldsymbol{\mu}^t) \tag{2.35}$$

$$\boldsymbol{\Sigma}^{t+1} = \boldsymbol{\Sigma}^t + \gamma^{t+1}\left[(\boldsymbol{X}^{t+1} - \boldsymbol{\mu}^t)(\boldsymbol{X}^{t+1} - \boldsymbol{\mu}^t)^T - \boldsymbol{\Sigma}^t\right], \tag{2.36}$$

where γ is a decreasing parameter which provide the Markov chain property described in the beginning of chapter 2.3. The details of γ^t, to ensure an irreducible and aperiodic Markov chain can be found in Roberts and Rosenthal (2007) and Atchadé et al. (2011). It is noted that $\lim_{t->\infty} \gamma^t = 0$, while not necessary bounded for $\sum_{t=1}^{\infty} \gamma = \infty$. Repeated trails concluded that $\gamma^t = 0.5\exp(t/\tau)$ was a sufficient choice, where $\tau = 0.1 \cdot N/\log(10)$ and N is the predefined realization length. The γ is constructed to be $1/20$ at $t = 0.1 \cdot N$.

As described above an adaptive λ^t can be beneficial. By using

$$\log(\lambda^{t+1}) = \log(\lambda^t) + \gamma^{t+1}\left(R(\boldsymbol{X}^*, \boldsymbol{X}^t) - a\right), \tag{2.37}$$

the series acceptance rate will converge towards a (Givens and Hoeting, 2013, p. 248).

A more detailed description of the adaptive metropolis algorithm can be found in (Givens and Hoeting, 2013, p. 247).

2.3.5 Applying Markov Chain Monte Carlo to Peak Over Threshold

In chapter 2.2, the two equation (2.28) and (2.30) construct the basis for the blocking Gibbs sampler

$$\xi^{t+1}, \phi^{t+1}|\cdot \sim c \cdot f_{\boldsymbol{y}|\xi,\sigma}\left(\boldsymbol{y}|\xi, \exp(\phi)\right) f_\xi(\xi) f_\phi(\phi)$$

$$\psi^{t+1}|\cdot \sim \text{BETA}(k+1, N-k+1),$$

where the parameters and functions are described in chapter 2.2. After the Markov chain sampling is complete, the transformation $\sigma = \exp(\phi)$ ensure correct parameter for the Monte

Carlo simulations. Sampling from ψ is straight forward since it is simply realizations of the Beta distribution, while ξ and σ are more complex and cannot directly be sampled. The challenge of calculating the computationally heavy c for each iteration favors the implementation of the Metropolis-Hastings algorithm.

The algorithm independency of the user for tuning and improved convergence speed makes the adaptive Metropolis-Hasting algorithm, described in chapter 2.3.4, favorable for ξ, σ. The posterior distribution often results in extremely small values, which in some cases could get disrupted by the violation of the smallest floating number for the software. To account for this, the logarithm of equation (2.34) is used. The resulting log Metropolis ratio becomes,

$$\begin{aligned}\log\left[R(\boldsymbol{X}^*, \boldsymbol{X}^t)\right] &= \log\left[f_{\xi,\phi|\boldsymbol{y}}(\xi^*, \phi^*|\boldsymbol{y})\right] - \log\left[f_{\xi,\phi|\boldsymbol{y}}(\xi^t, \phi^t|\boldsymbol{y})\right] \\ &= \log\left[f_{\boldsymbol{y}|\xi,\sigma}\left(\boldsymbol{y}|\xi^*, \exp(\phi^*)\right)\right] - \log\left[f_{\boldsymbol{y}|\xi,\sigma}\left(\boldsymbol{y}|\xi^t, \exp(\phi^t)\right)\right] + \\ &\quad \log\left[f_\xi(\xi^*)\right] + \log\left[f_\phi(\phi^*)\right] - \log\left[f_\xi(\xi^t)\right] - \log\left[f_\phi(\phi^t)\right],\end{aligned} \tag{2.38}$$

where equation (2.26) gives,

$$\log\left[f_{\boldsymbol{y}|\xi,\sigma}\left(\boldsymbol{y}|\xi, \exp(\phi)\right)\right] = \begin{cases} -n\phi - \exp(-\phi)\sum_{i=1}^n y_i, & \xi = 0, \\ -n\phi - \left(1 + \frac{1}{\xi}\right)\sum_{i=1}^n \log\left(1 + \xi\exp(-\phi)y_i\right), & \xi \neq 0. \end{cases} \tag{2.39}$$

After inserting mean and variance of the priors from chapter 2.2, the remaining parts reduces to

$$\log\left[f_\xi(\xi^*)\right] + \log\left[f_\phi(\phi^*)\right] - \log\left[f_\xi(\xi^t)\right] - \log\left[f_\phi(\phi^t)\right] = -\frac{(\xi^*)^2 - (\xi^t)^2}{200} - \frac{(\phi^*)^2 - (\phi^t)^2}{2\cdot 10^4}. \tag{2.40}$$

For the equations, t indicate the parameter iteration number, * indicate the proposed parameter value to be evaluated, ξ and σ are GPD parameters where $\phi = \log(\sigma)$ and $\boldsymbol{y}$ is a vector containing the observed data of size n.

The remaining construction of the AMCMC simulator is as described above in chapter 2.3.4. The result is a hybrid Gibbs AMCMC simulator for the POT method. The Markov chain is valid since both Gibbs steps are irreducible and aperiodic. Irreducible because each sampler within their restricted range can sample any realization with probability larger than zero, from any state. Aperiodic since both Gibbs steps can return to their state in a single iteration,

with probability larger than zero.

To limit the possibility of the adaptive Metropolis-Hastings algorithm getting stuck in a slowly converging area before reaching the target distribution, multiple independent MCMC simulations with different starting values for ξ^0 and ϕ^0 are used. After convergence the latest value of ξ^t and ϕ^t between the MCMC simulations with the highest $f_{\boldsymbol{y}|\xi,\sigma}(\boldsymbol{y}|\xi^t, \exp(\phi^t))f(\xi^t)f(\phi^t)$ are selected for future realization generation. All points generated prior to this point for each starting value are discarded as burn-in.

Estimation and credible interval can simply be added through Monte Carlo simulation. For a parameter realization of size T, after burn-in is removed and the effective sample size is sufficient, equation (2.24) is estimated as

$$\hat{\Pr}(Z > z|\boldsymbol{y}) = \frac{1}{T}\sum_{t=1}^{T} \Pr(Z > z|\xi^t, \sigma^t, \psi^t). \tag{2.41}$$

The estimation is unbiased since $(\xi^t, \sigma^t, \psi^t) \sim f(\xi, \sigma, \psi|\boldsymbol{y})$. The $100 \cdot \alpha\%$ credible interval for a specific $z = s$ can be added to the $\hat{\Pr}(Z > s|\boldsymbol{y})$ estimation by sorting the result of $\Pr(Z > s|\xi^t, \sigma^t, \psi^t)$ for every realization, and selecting lower and upper limits for where $100 \cdot \alpha\%$ of the results are contained. The narrowest interval is chosen for this work and is called the highest posterior density interval. Example of the use of such interval can be seen in figure 4.3, where the credible interval has been estimated on multiple z values for the MCMC method. The complete `R` and C++ code for calculating the AMCMC for the POT method, can be found in appendix B.

2.3.6 Multivariate Random Normal Generator

Most of the coding for this work was done in `R`. Since the MCMC accuracy increase with the numbers of iteration generated, parts of the AMCMC algorithm was coded in C++, through the Rcpp package by Eddelbuettel et al. (2016), for speed optimization. The AMCMC for ξ and ϕ uses a bivariate random normal generator, but this is not natively supported in C++. Since no additional packages tested were satisfactory for the purpose, the bivariate random normal generator was constructed.

The Box-Muller transformation, see Box and Muller (1958), states that for two indepen-

dent random variables $U_1, U_2 \sim UNIF(0,1)$, the transformation

$$Z_1 = \sqrt{-2\log(U_1)}\,\cos(2\pi U_2)$$
$$Z_2 = \sqrt{-2\log(U_1)}\,\sin(2\pi U_2)$$

results in Z_1 and Z_2 to be independent and standard normally distributed. Combined in a vector $\boldsymbol{z} = [Z_1, Z_2]^T$, where T is the transpose, the bivariate random normal $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be generated by

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{A}\boldsymbol{z}, \tag{2.42}$$

where $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^T$. Here $\boldsymbol{A}$ is chosen as the Cholesky decomposition of $\boldsymbol{\Sigma}$. The C++ code can be found in appendix B.

2.4 Forecasting Extreme

Two methods were used for prediction and forecasting of future extreme.

2.4.1 Value at Risk

Value at risk (VaR) for confidence α is defined as the smallest value l where the loss L of the portfolio will exceed with probability α, over a given time period h

$$\Pr(L_h \geq l_{h,\alpha}) = \alpha. \tag{2.43}$$

As this work only considers one day VaR the simplified notation $\mathrm{VaR}_\alpha = l_\alpha$ is used.

By inverting equation (2.16), VaR of the ACER method for a given k is calculated as

$$\mathrm{VaR}_\alpha = b + \left[\frac{1}{a\xi}\left(\frac{q}{\alpha}\right)^{\xi} - 1\right]^{1/c}. \tag{2.44}$$

For the POT MCMC method, equation (2.25) is inverted, for which the VaR follows

$$\mathrm{VaR}_\alpha^t = u + \frac{\sigma^t}{\xi^t}\left[\left(\frac{\psi^t}{\alpha}\right)^{\xi^t} - 1\right], \tag{2.45}$$

where t indicates the sampled realization number from the MCMC. As the reasoning for equation (2.41), the estimated VaR becomes $\mathrm{VaR}_\alpha = 1/T\sum_{t=1}^{T}\mathrm{VaR}_\alpha^t$, where T is the total num-

ber of realizations generated after burn-in is removed.

Dependent if the portfolio is on buy or sell, the value at risk make sense for both increase and decrease of daily return respectively. For VaR on decreasing data, the extreme of the negative dataset $-X_1, \ldots, -X_n$ is analyzed instead.

2.4.2 Forecasting Prediction Interval of Extreme

The VaR gives an approach for testing how well the ACER and POT MCMC methods estimates the underlying distribution. The VaR estimation does not reflect how well the ACER and POT MCMC captures the estimation variability. By analyzing the maximum of a future time series, a method for testing variability is developed.

As stated earlier in chapter 2.1.1, for time series with no trend, the most extreme events are approximately i.i.d, thus at the tails, equation (2.1) can be approximated as (2.2). The estimated cumulative distribution $\hat{F}(z)$ can thereby be used for estimating the maximum of a future long time series by

$$\Pr(M_n < z) \approx \left[\hat{F}(z)\right]^n, \tag{2.46}$$

where n is the number of points in the time series, as in chapter 4.2. Both the ACER and POT MCMC method are developed for $\hat{\Pr}(X > z) = 1 - \hat{F}(z)$, see equation (2.16) and (2.25) respectively. The $100 \cdot \alpha\%$ Prediction Interval (PI) of the most extreme of n points, or nth extreme, can be found by

$$\alpha = \Pr(m_{n,p_1} \leq M_n \leq m_{n,p_2}), \tag{2.47}$$

where $\Pr(M_n < m_{n,p_1}) = p_1$, $\Pr(M_n < m_{n,p_2}) = p_2$, $p_2 - p_1 = \alpha$, and both p_1 and p_2 is between 0 and 1. Using equation (2.46), the value of $m_{n,p}$ for a given n and p can be estimated by inverting (2.16), resulting in

$$\hat{m}_{n,p} = b + \left[\frac{1}{a\xi}\left(\frac{q}{1-p^{1/n}}\right)^{\xi} - 1\right]^{1/c}. \tag{2.48}$$

for the ACER method. By computing $\hat{m}_{n,p_1}$ and $\hat{m}_{n,p_2}$ for a variety of p_1, p_2 satisfying $\alpha = p_2 - p_1$, the highest posterior density interval is chosen as PI.

The PI for the POT MCMC method is calculated by generating realizations of the nth extreme z_n, using various parameter realizations generated by the MCMC method. Values of the estimated z_n can be generated using the probability integral transform approach, where

$\Pr(X < x) \sim \text{UNIF}(0,1)$, combined with equation (2.25) and (2.46), this gives

$$\hat{z}_n^i = u + \frac{\sigma^t}{\xi^t}\left[\left(\frac{\psi^t}{1-y_i^{1/n}}\right)^{\xi^t} - 1\right], \tag{2.49}$$

where $y_i \sim \text{UNIF}(0,1)$, i is the ith realization of $\hat{z}_n$ and t is the parameter realization number. For a high number of realizations, the $100 \cdot \alpha\%$ highest posterior density interval is chosen by sorting $\hat{z}_n^i$ by values and selecting the narrowest continous interval which holds $100 \cdot \alpha\%$ of the data. The POT MCMC PI also captures the parameter variation. Unfortunately this is not the case for the PI generated by the ACER method and therefore probably will result in a too narrow PI.

2.5 Test and Evaluation

2.5.1 Ljung–Box test

The Ljung-Box test is a method for test if data in a time series is independently distributed. The test is given by

$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k}, \tag{2.50}$$

were $\hat{\rho}_k$ is the sample autocorrelation function for lag k, n is the data length and h is the upper limits of the numbers of lags included in the test. For testing purposes $Q \sim \chi^2$ with $h - p$ degrees of freedom, where p represent the degrees of freedom lost in the filtering method. For non-filtered data $p = 0$. The filtering method used in this work is the ARMA-APARCH described below.

2.5.2 Likelihood Ratio Test

Likelihood ratio test is a method of comparing the goodness of fit for different models. The test required a null model $\boldsymbol{\theta_0}$, and an alternative model $\boldsymbol{\theta_a}$, where the null model is a nested subset of the alternative. The test statistics is approximately chi-square distributed, with $df_a - df_0$ degrees of freedom, where df_a and df_0 is the alternative and null models number of free parameters respectively. The log likelihood ratio test statistics have the following relationship

$$\chi^2_{(df_a - df_0)} \sim -2\log\left[\frac{f(\boldsymbol{x}|\boldsymbol{\theta_0})}{f(\boldsymbol{x}|\boldsymbol{\theta_a})}\right] \tag{2.51}$$

where $f(\boldsymbol{x}|\boldsymbol{\theta})$ is the likelihood, as seen in equation (2.22) and χ^2 is the chi-square distribution.

2.5.3 Evaluating Forecasts

The out of sample VaR and the max of a future nth event prediction interval, is evaluated by the test of Kupiec (1995) and Christoffersen (1998). These tests can be developed from the likelihood ratio test statistics described above, but are only presented here. Both test define an indication variable I_t as

$$I_t = \begin{cases} 1, & \text{if} y_t \in \left[L_{t|t-1}(p), U_{t|t-1}(p)\right], \\ 0, & \text{if} y_t \notin \left[L_{t|t-1}(p), U_{t|t-1}(p)\right], \end{cases} \tag{2.52}$$

for $1 \leq t \leq n$, where $L_{t|t-1}(p)$ and $U_{t|t-1}(p)$ is the lower and upper prediction limits respectively for time t conditioned on the previous data and probability p, and y_t is the corresponding observed data.

The Kupiec (1995) tests the unconditional coverage of the model, by testing the hypothesis that $E[I_t] = p$ against the alternative $E[I_t] \neq p$. The test statistics is calculated by

$$\text{LR}_{\text{UC}} = -2\log\left[\frac{(1-p)^{n_0} p^{n_1}}{(1-\hat{\pi})^{n_0}\hat{\pi}^{n_1}}\right], \tag{2.53}$$

where $n_1 = \sum_{i=t}^{n} I_t$, $n_0 = n - n_1$ and $\hat{\pi} = n_1/n$. The test statistics is proportional to the chi-square distribution $\text{LR}_{\text{UC}} \sim \chi_1^2$ with 1 degree of freedom, and is used for hypotheses evaluation.

The Christoffersen (1998) also tests the conditional coverage of the model, by testing the hypothesis that $E[I_t] = p$ against the alternative $E[I_t] \neq p$. In contrast with the above test, the Christoffersen (1998) accounts for dependency. The test statistics is calculated by

$$\text{LR}_{\text{CC}} = -2\log\left[\frac{(1-p)^{n_0} p^{n_1}}{(1-\hat{\pi}_{01})^{n_{00}}\hat{\pi}_{01}^{n_{01}}(1-\hat{\pi}_{11})^{n_{10}}\hat{\pi}_{11}^{n_{11}}}\right], \tag{2.54}$$

where n_{ij} is the numbers of I_t going from i to j and $\hat{\pi}_{ij} = n_{ij}/n_i$. The test statistics is proportional to the chi-square distribution $\text{LR}_{\text{CC}} \sim \chi_2^2$ with 2 degree of freedom, and is used for hypotheses evaluation.

For VaR evaluation, the upper limit $U_{t|t-1}(p)$ is set to infinity, which result in $L_{t|t-1}(p) =$

VaR_p, where VaR_p is calculated using the previous observations.

2.6 ARMA-APARCH

The Autoregressive Moving Average (ARMA) model is a method of analyzing time series. The ARMA(p,q) model for a time series $Z_1,\ldots,Z_T$, where $\dot{Z}_t = Z_t - \mu$ and μ is the intercept of Z, is given by

$$\dot{Z}_t = \sum_{i=1}^{p} \phi_i \dot{Z}_{t-i} \sum_{j=1}^{q} \theta_j e_{t-j} + e_t, \tag{2.55}$$

where ϕ is the Autoregressive (AR) parameters, θ is the Moving Average (MA) parameters and e is the error term. The p indicates the highest AR order while q indicates the highest MA order in the model. For $q = 0$ or $p = 0$ the ARMA(p,q) model is referred to as AR(p) or MA(q) respectively.

In situations where the error term is heteroscedastic, an Asymmetric Power Autoregressive Conditional Heteroscedasticity (APARCH) model can be included. For the time series above, the APARCH(k,l) model is

$$\sigma_t^{\delta} = \omega + \sum_{k=1}^{r} \alpha_k \left(|e_{t-k}| - \gamma_k e_{t-k}\right)^{\delta} + \sum_{l=1}^{s} \beta_l (\sigma_{t-l})^{\delta}, \tag{2.56}$$

where α, β, ω, γ and δ is the parameters while $e_t = \sigma_t \epsilon_t$, r is the highest α or $gamma$ order, s is the highest β order and ϵ_t is a homoscedastic standardized error term. The APARCH model equals the Autoregressive Conditional Heteroscedasticity (ARCH) for $\delta = 2,\ \gamma = 0,\ \beta = 0$, the Generalized Autoregressive Conditional Heteroscedasticity (GARCH) for $\delta = 2,\ \gamma = 0$ and the Glosten Jagannathan Runkle (GJR) GARCH for $\delta = 2$. The APARCH model has shown to work well for fat tails, excess kurtosis and leverage effects. For the data analyzed in this work, the standardized error term ϵ_t is assumed to follow a skewed student t-distribution. Thus $\epsilon_t \sim \text{SKEW}(\mu^* = 0, \sigma^* = 1, \nu, \xi)$, where μ^* is location, σ^* is scale, ν is shape and ξ is the skewness parameter of the skewed student t-distribution. The theory behind the skewed student t-distribution is not presented here, as the fGarch `R` package by Wuertz et al. (2013) handles it automatically, but a detailed description can be found in Fernandez and Steel (1998).

Chapter 3

Data

The following chapter introduce the different types of data analyzed. Synthetic data are generated an analyzed, for better control of behavior and result verification of the ACER and POT MCMC method before the methods are applied for a variety of commodity times series.

3.1 Synthetic Data

Two synthetic data series are generated. Both attempts to test the methods on different aspects which are important for real life daily return commodity analysis. The first is the thick tail Pareto distribution, since commodity data have shown signs of fat tail distribution, as the study by Aloui and Mabrouk (2010) presented. The Pareto distribution can generate i.i.d. data which are easy to control and where exact analytic inference can be achieved.

The second, numerically generates a time series with approximately the same distribution characteristics as the return of crude oil commodity data. Since it is a numeric method, the data can be generated as large as desired, and much larger than is practical achievable in real life. Thus more accurate test result is achieved compared to limited real life data.

3.1.1 Pareto Distribution

The Pareto distribution has the following cumulative distribution function

$$\Pr(X < x) = \begin{cases} 1 - \frac{1}{x^{\beta}} & \text{if } x > 1 \\ 0 & \text{if } x < 1 \end{cases} \tag{3.1}$$

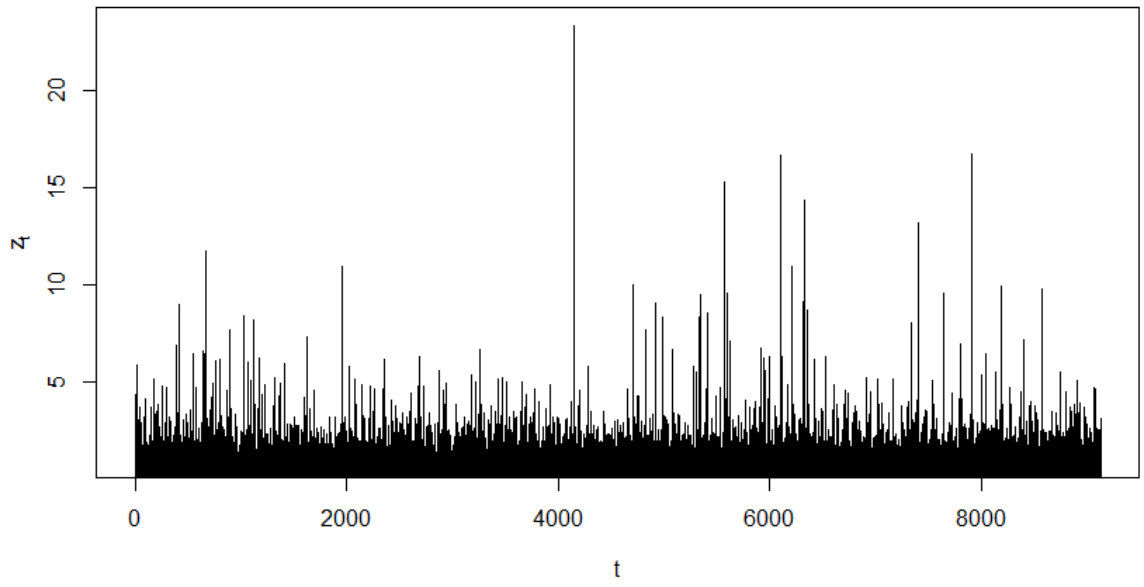

Figure 3.1: Generated Pareto distributed points with 9125 realization using $\beta = 3$. The i axis represent time in days, while x_i is the realized data point for that day.

where β is a shape parameter. A cumulative distribution function is said to have fat tail if $\Pr(X > x) \sim x^{-\beta}$, as $x \to \infty$, for $\beta > 0$. By the definition, the Pareto distribution clearly follow the property of fat tail distribution. Using the probability integral transform approach, a data point x_i is generated by

$$x_i = \frac{1}{u_i^{1/\beta}}, \tag{3.2}$$

where $u_i \sim \text{UNIF}(0,1)$.

A practical data size could come from having a daily return each day for 25 years. Without considering leap year, roll-over returns, and weekend, that corresponds in a 25 times 365 data series. Multiple data series of that size was generated using $1 < \beta < 5$.

Figure 3.1 shows a generated Pareto distributed time series for $\beta = 3$, where the size of realizations range from 1.000 to 23.384.

3.1.2 ARMA-APARCH Generated Data

Since the Pareto distribution above generated i.i.d. data points, it seems logical to simulate dependent and homoscedastic data to reveal any performance difference on that aspect of the methods. This work focus on the ACER and POT MCMC methods ability to capture the

tail effect of commodity data, hence the ability to generating data with close to the same behaviors as real life commodity is desired.

Table 3.1: Estimated AR(3) - APARCH(1,1) parameters and p-values for the crude oil daily return (CL1). The parameter notation is as described in chapter 2.6.

	Estimate	p-value
μ	$4.13 \cdot 10^{-4}$	0.134
ϕ_1	$-2.55 \cdot 10^{-2}$	$9.87 \cdot 10^{-2}$
ϕ_2	$-4.34 \cdot 10^{-2}$	$2.45 \cdot 10^{-3}$
ϕ_3	$-1.72 \cdot 10^{-2}$	0.228
ω	$9.77 \cdot 10^{-5}$	$2.21 \cdot 10^{-3}$
α_1	$5.55 \cdot 10^{-2}$	$< 2 \cdot 10^{-16}$
γ_1	$2.33 \cdot 10^{-1}$	$2.61 \cdot 10^{-3}$
β_1	$9.50 \cdot 10^{-1}$	$< 2 \cdot 10^{-16}$
δ	1.13	$2.07 \cdot 10^{-7}$
ξ	$9.62 \cdot 10^{-1}$	$< 2 \cdot 10^{-16}$
ν	7.34	$< 2 \cdot 10^{-16}$

It was suggested by Giot and Laurent (2003), that the return of commodity time series approximately follows AR(3)-APARCH(1,1), with the standardize error term following a skewed student t distribution. Using the observed crude oil daily return (CL1) described below in chapter 3.2, the parameters of AR(3)-APARCH(1,1) with p-values is presented in table 3.1. All parameters except μ and ϕ_3 are significantly different from zero. The likelihood-ratio test statistics of multiple reduced models all performed significantly worse than the AR(3)-APARCH(1,1).

A generated time series from the AR(3)- APARCH(1,1) model with parameters as in table 3.1, can be seen in figure 3.2. The generated AR(3)- APARCH(1,1) heteroscedastic behavior seems comparable to the real commodity data sets described below.

3.2 Commodity Data

The daily return commodity data sets was provided by Sjur Westgaard. The daily return was calculated by

$$r_t = \frac{x_t - x_{t-1}}{x_t} \tag{3.3}$$

where r_t is daily return and x_t is the value of the commodity for day t. The data often experiences extreme changes between contract expiration and new contracts, these roll-over effects are accounted for by removing the specific daily return. For simplicity, any missing

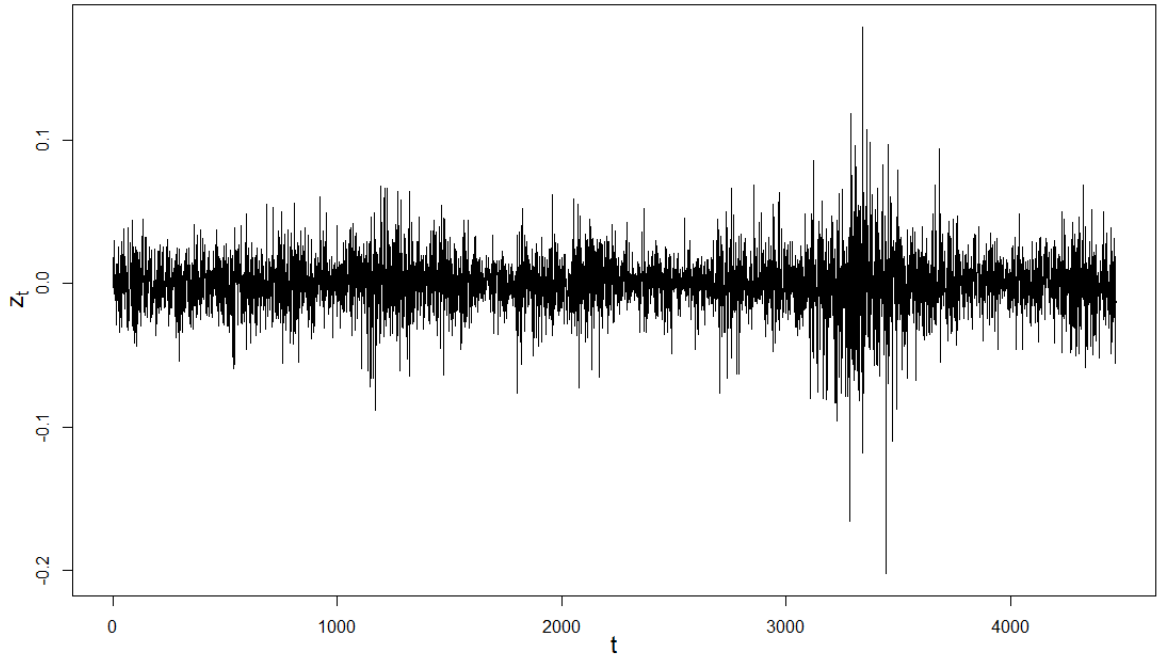

Figure 3.2: Generated AR(3)- APARCH(1,1) distributed points of length 4 469. The t axis represent time in days, while z_t is the realized data point for that day.

daily return for a commodity is also removed for other commodities. The data removal result in 10 commodity data series, with 4 469 daily return each spanning from August 3rd 1992 to November 12th 2013. A plot of each of the daily return series can be seen in figure 3.3. The plot reveals the non-stationarity in variance for each of the data series. Some of the increased variance clusters spans over multiple years.

Descriptive statistics for each of the commodities can be found in table 3.2. Except wheat (W1), soybean (S1) and Soyabean oil (BO1), the data independency hypothesis is rejected for the Ljung-box test statistic at a 5% significance level.

In an effect to accommodate for the non-stationarity in variance, each data series was filtered using the AR(3)- APARCH(1,1) model with skewed student t-distributed errors, as was described above for the crude oil data. The filtered standardized residuals from each of the commodity data series can be seen in figure 3.4, with corresponding descriptive statistics in table 3.2 noted F. for filtered. For each of the filtered data, the Ljung-box test was not rejected on a 5% significance level.

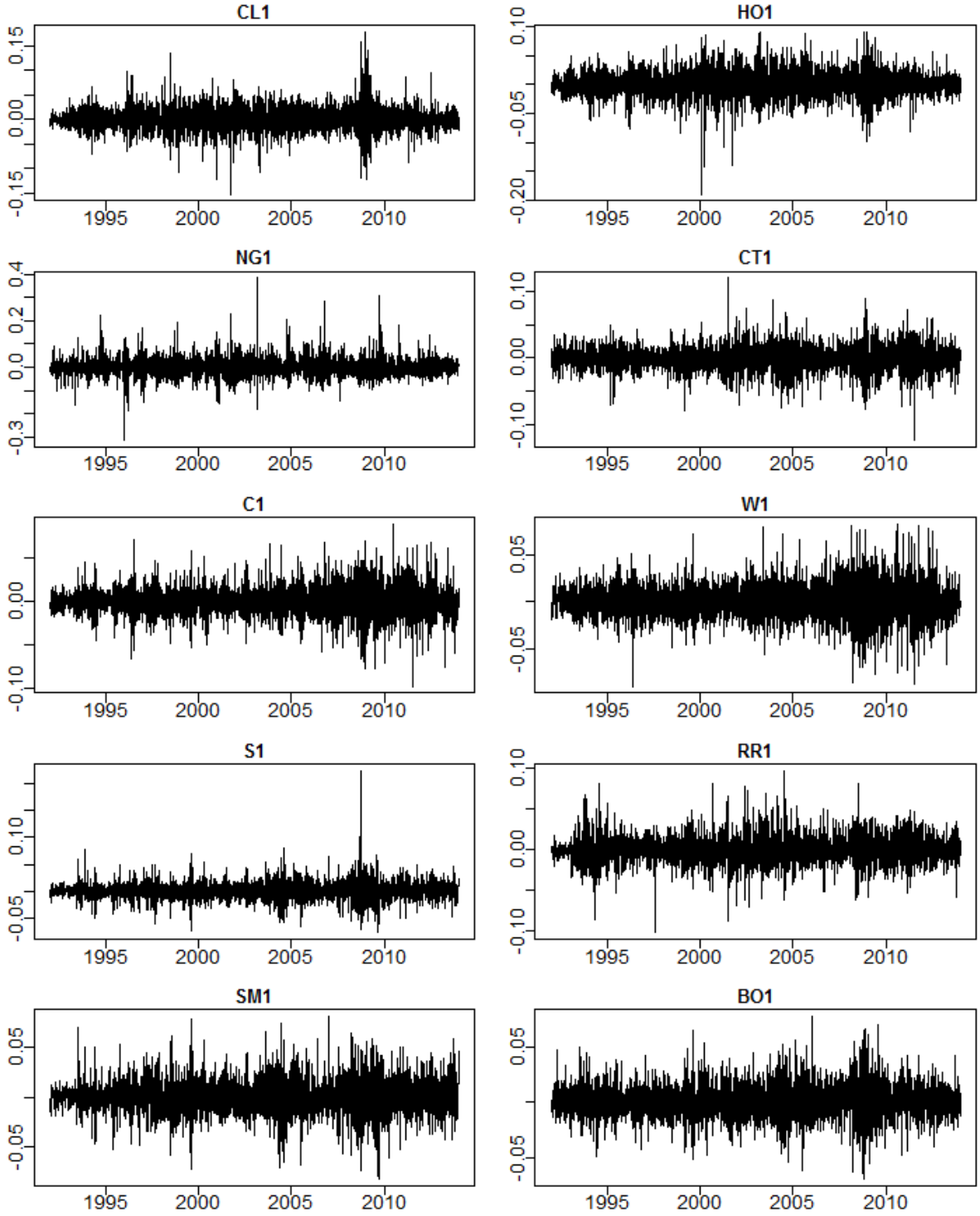

Figure 3.3: Plots of daily returns against time in years for each of the commodity data series.

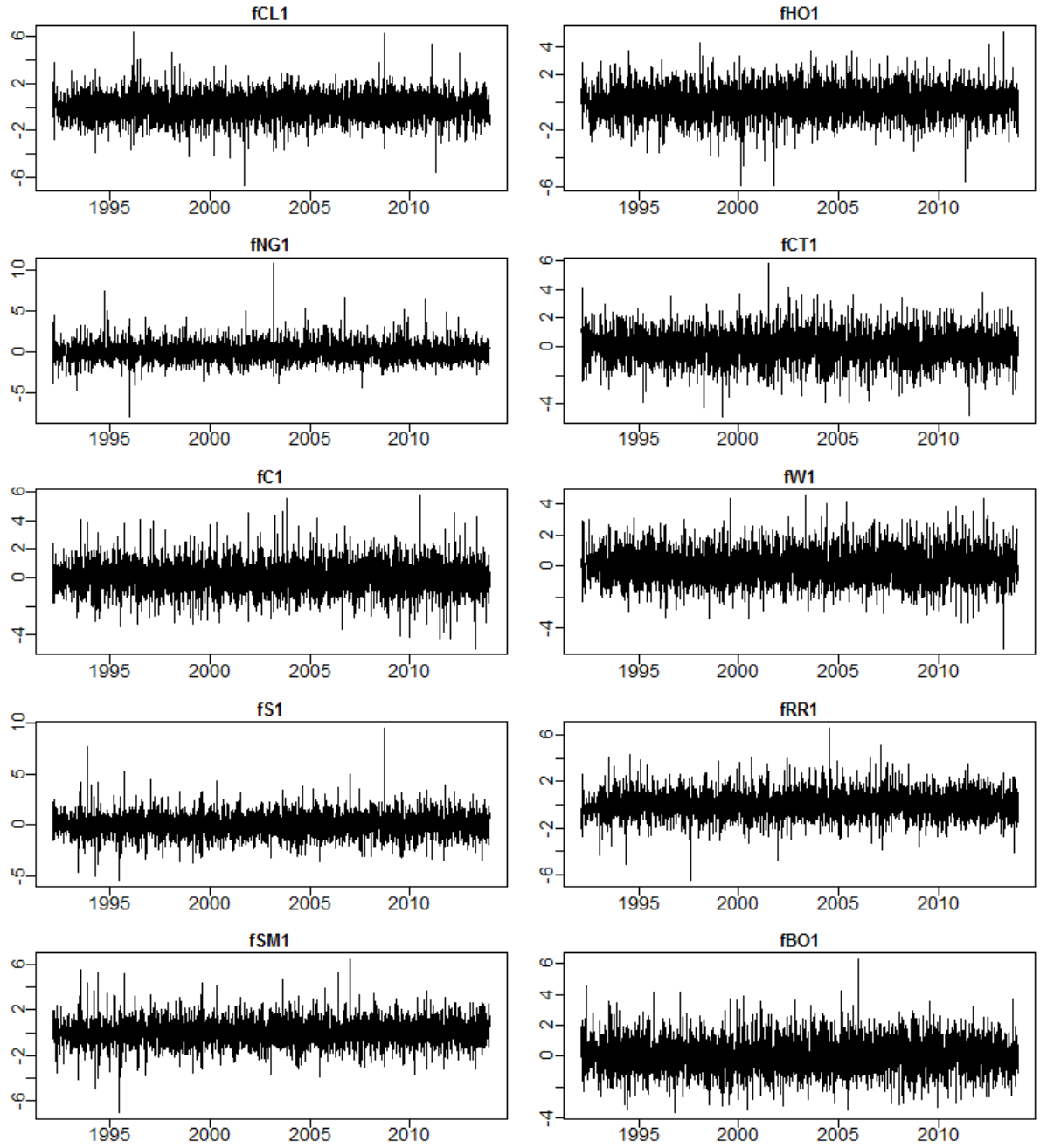

Figure 3.4: Plots of the standardized residuals against time in years for each of the AR(3)-APARCH(1,1) filtered data series.

Table 3.2: Descriptive statistics for the commodities data and standardized residuals of the AR(3)- APARCH(1, 1) filtered commodities.

Symbol	Marked	Mean	SD	$Q(0.05)$	$Q(0.95)$	Skew	Kurtosis
CL1	Crude Oil	0.00	0.02	−0.04	0.03	0.14	7.98
HO1	Heating Oil	0.00	0.02	−0.03	0.03	−0.24	6.45
NG1	Natural Gas	0.00	0.04	−0.05	0.06	0.69	11.51
CT1	Cotton	0.00	0.02	−0.03	0.03	−0.06	5.73
C1	Corn	0.00	0.02	−0.03	0.03	0.05	5.61
W1	Wheat	0.00	0.02	−0.03	0.03	0.08	5.11
S1	Soybean	0.00	0.02	−0.02	0.02	0.61	16.24
RR1	Rough Rice	0.00	0.02	−0.03	0.03	0.12	5.75
SM1	Soyabean Mean	0.00	0.02	−0.02	0.03	0.00	5.28
BO1	Soyabean Oil	0.00	0.01	−0.02	0.02	0.19	5.00
fCL1	F. Crude Oil	0.01	1.01	−1.65	1.61	0.00	2.37
fHO1	F. Heating Oil	0.00	1.00	−1.60	1.63	−0.05	1.53
fNG1	F. Natural Gas	0.01	1.03	−1.49	1.65	0.77	6.28
fCT1	F. Cotton	0.00	1.02	−1.68	1.61	−0.47	5.31
fC1	F. Corn	0.00	1.03	−1.57	1.61	−0.16	4.74
fW1	F. Wheat	0.00	1.00	−1.54	1.65	0.17	0.79
fS1	F. Soybean	0.01	1.07	−1.60	1.57	0.26	3.94
fRR1	F. Rough rice	0.00	1.01	−1.61	1.62	0.13	2.17
fSM1	F. Soyabean mean	0.01	1.07	−1.54	1.65	0.14	2.62
fBO1	F. Soyabean oil	0.00	1.00	−1.56	1.69	0.26	1.20

Chapter 4

Analysis and Results

The following chapter contains the analysis of the data presented in chapter 3. First is the analysis of the two computer generated synthetic data, before the methods are applied to the real life commodity data and corresponding standardized residuals.

As described in chapter 2.3.4, the POT MCMC method optimal acceptance rate for a single dimensional normal target distribution is 44% while it is 23.4% for higher dimensional multivariate normal target distribution. The parameters ξ and ϕ probably does not follow a bivariate normal target distribution, and they are only two dimensional, hence the optimal acceptance rate is unknown. Calculating the effective sample size for multiple different MCMC simulations on different data sets, using a variety of acceptance rate between 20% and 50%, shows a trend suggesting $30\% \leq a \leq 40\%$. For the rest of this work a is set to 0.35, which will converge the AMCMC towards an acceptance rate of 35%, by the theory in chapter 2.3.4.

4.1 Analysis of Syntetic data

The goal of this section is to use the ACER and POT MCMC method to analyze controlled data for better conformation of prediction, test reliability and performance difference. The Pareto distribution is i.i.d. while the simulated AR(3)-APARCH(1, 1) is dependent by the Ljung-box test statistic.

4.1.1 Pareto Distirbution

An advantage of the Pareto distribution is that the theoretical distribution is known, hence the exact analytical values can be achieved even for extreme events. For testing, the yearly, decadal, centennial and millennial event or in probability 1/365, 1/3650, 1/36500 and 1/3650000 respectively, are estimated and compared with the exact values.

Given the definition of VaR equation (2.43), the exact VaR for the Pareto distribution can be found by

$$\mathrm{VaR}_\alpha = \frac{1}{\alpha^{1/\beta}}.$$

For the estimated PI, the exact probability for a future nth maximum to arrive whiten that limit is developed by combining equation (2.2) and (3.1), which result in

$$\Pr\left(m_n^{(-)} \leq M_n \leq m_n^{(+)}\right) = \left(1 - m_n^{(+)-\beta}\right)^n - \left(1 - m_n^{(-)-\beta}\right)^n,$$

where $m_n^{(+)}$ and $m_n^{(-)}$ is the upper and lower PI limits respectively.

The Pareto data from figure 3.1, with $\beta = 3$ is analyzed for a walkthrough of how the methods are used, and result interpretation. The exact VaR for each case are represented in table 4.1.

Table 4.1: The exact VaR for each event with the corresponding probability α, for the Pareto distribution with $\beta = 3$.

Event	α	VaR_α
Yearly	1/365	7.147
Decadal	1/3650	15.397
Centennial	1/36500	33.171
Millennial	1/365000	71.466

Since the data are i.i.d., the ACER methods will use $k = 1$, and the POT MCMC method will not need any declustering setting or threshold analysis. In theory the point for where regular tail behavior starts for the ACER method, and the threshold for the MCMC method could be selected to 1, which would make use of the entire data set without removing lower data. In real life such a situation is unrealistically, thus both is selected to 1.5. Consequently, out of the total 9 125 Pareto distributed points, 2 728 exceeds 1.5 and is used for parameter estimation.

The POT MCMC method is set to generate 100 000 realizations. In figure 4.1 the first 3 000 realizations of ξ and σ is shown for starting conditions set to $[\xi^0, \sigma^0] = [1, 1]$. The plot

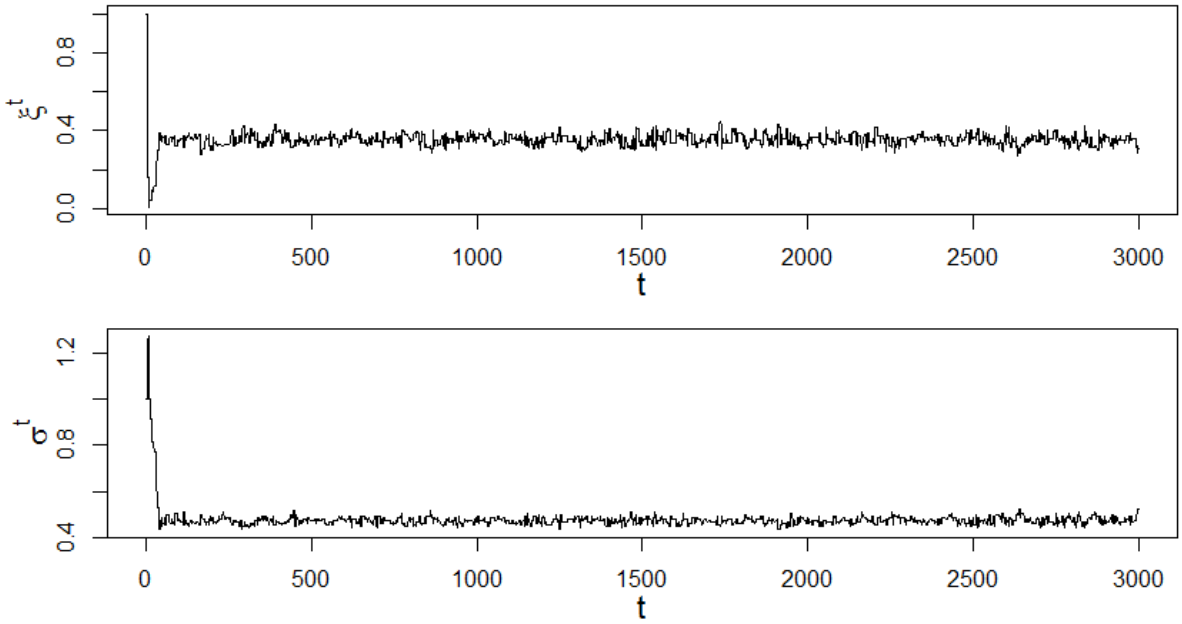

Figure 4.1: The first 3 000 realizations of the POT MCMC method for ξ and σ.

converges within a few numbers of steps, while the variance of the points is adapting for more iterations. Points generated after convergence, does follow the correct distribution, the variance only affects the effective sample size, even though it has not settled in yet. Data between 1st and 500th realization is selected as burn-in and is discarded for future analysis, resulting in an effective sample size of 13 580 for ξ and 14 746 for σ. The realized distribution of ξ, σ and β for the POT MCMC method can be seen in figure 4.2

For the ACER method, the estimates, together with the upper and lower confidence limits can be seen in table 4.2, with corresponding lines plotted in figure 4.3.

Table 4.2: The estimated ACER parameters together with the upper and lower 95% confidence interval line parameters.

	a	b	c	q	ξ
Upper 95% CI line	2.824	1.029	1.626	0.577	0.769
Estimated line	2.756	1.000	1.356	0.690	0.545
Lower 95% CI lien	3.105	1.000	0.652	1.747	0.110

Comparison of the ACER and POT MCMC estimated probability functions with the exact Pareto distribution can be seen in figure 4.3. Defining loss as an increase in value, the plot can also be interpreted as α against VaR_α. It is noted that the estimates for the POT MCMC are closer to the real distribution, with slimmer confidence interval.

Using the theory from chapter 4.2 for multiple probabilities, a numerical prediction dis-

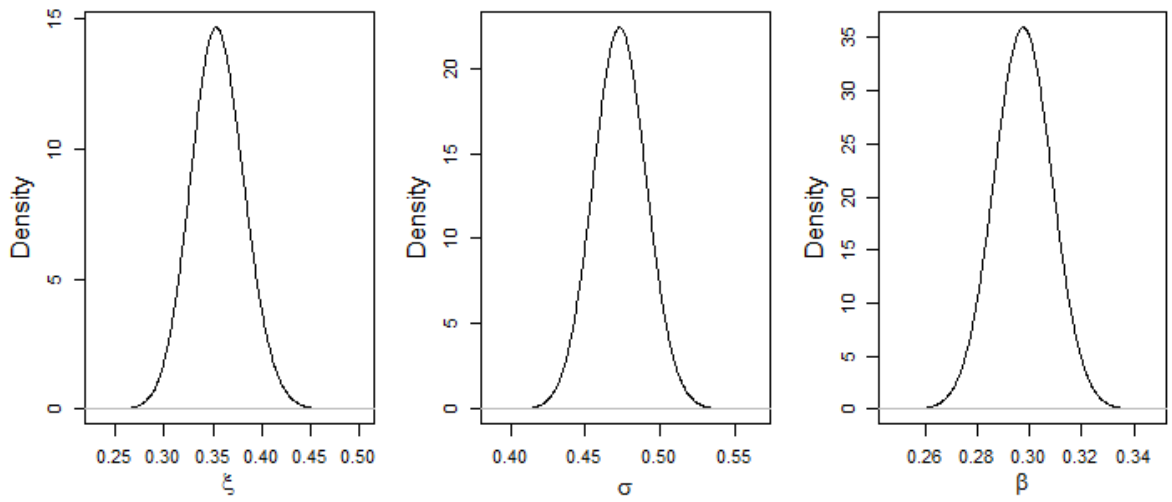

Figure 4.2: The estimated posterior density of the POT MCMC parameters ξ, σ and β.

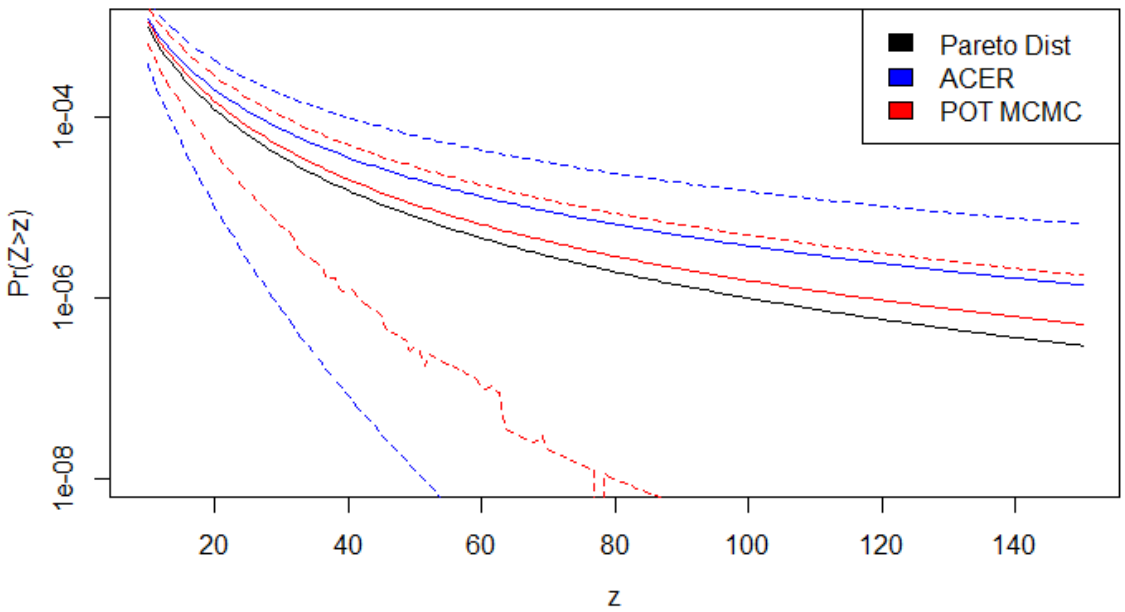

Figure 4.3: A plot visualizing the estimated ACER and POT MCMC probabilities, together with the exact Pareto distribution. The estimates are shown in solid lines while the upper and lower 95% confidence and credible interval are dashed.

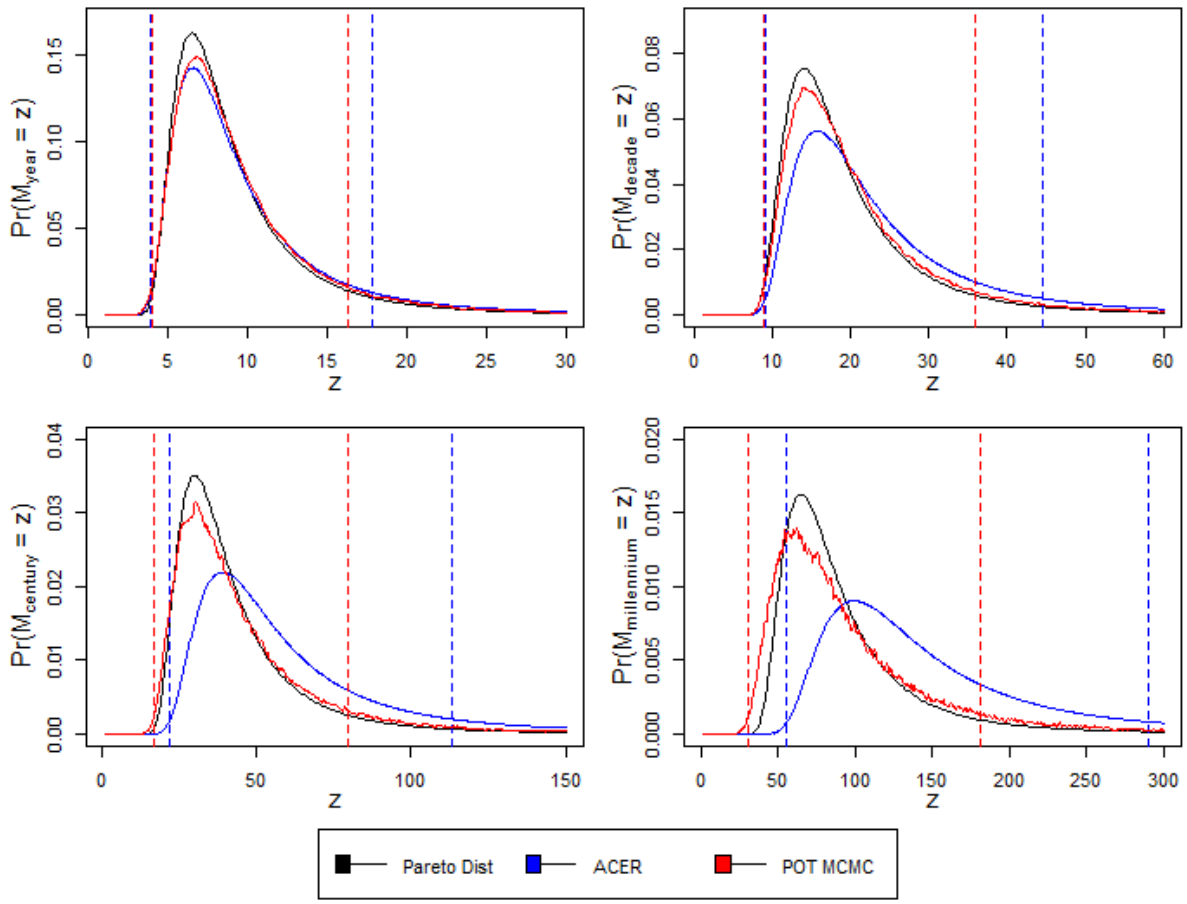

Figure 4.4: The estimated ACER and POT MCMC future year, decade, century and millennium predicted probability functions together with the true distribution. Solid line shows the probability distribution functions while dashed lines mark each methods 90% PI.

tribution for the maximum event of the upcoming year, decade, century and millennium is estimated for the two methods. Figure 4.4 shows the distribution of the exact Pareto prediction together with the estimated ACER and POT MCMC, with PI.

An approach for validating the methods PI, comes from calculating how much of the exact distribution was contained within the PI. See table 4.3 for the corresponding values. The table shows that both methods for each event predict reasonable close to 90% interval,

Table 4.3: The amount of the exact distribution contained in the 90% PI, with interval length in brackets, for the ACER and POT MCMC method.

	Yearly	Decadal	Centennial	Millennial
ACER	0.935 (13.9)	0.948 (34.8)	0.936 (87.58)	0.851 (220.9)
POT MCMC	0.920 (12.7)	0.928 (28.9)	0.941 (67.6)	0.954 (159.9)

while the POT MCMC method perform better with respect to interval width for this particular

case.

A better understanding of the performance difference for the two method is achieved by repeating the above analysis for multiple scenarios. The 9 125 points Pareto distribution start series is regenerating 100 times using a random β between 1 and 5. The point for where regular tail behavior starts z_1 for the ACER, and threshold u for the POT MCMC method are both set to the 0.33 quantile of the generated points. As described above, the ACER k is set to 1 while no declustering is applied for the POT MCMC. A summary of the result for the 100 data series can be found in table 4.4. The column name #Best VaR refers to the number of times the corresponding method VaR estimation falls closer to the exact VaR than the competing method. The #in CI refers to the number of times the exact VaR falls within the 95% ACER confidence or POT MCMC credible interval. The #Best CI is defined as the number of times the corresponding method confidence or credible interval is narrower than the competing method, while achieving the Best VaR estimation. As in table 4.3, the amount of the exact distribution contained within the 90% PI is calculated for each of the 100 data series. The resulting mean and standard deviation can be seen in table 4.4. The #Best PI is defined as the number of times the corresponding method PI is narrower than the competing method, while the amount of the exact distribution contained within the PI is higher.

Table 4.4: A summary of the combined result for the 100 generated Pareto distributed data series.

		Estimated VaR			90% Prediction Interval		
		#Best VaR	#in CI	#Best CI	Mean	SD	#Best PI
Yearly	ACER	44	99	0	0.90	0.03	0
	POT MCMC	56	95	56	0.90	0.02	0
Decadal	ACER	34	99	0	0.89	0.05	1
	POT MCMC	66	94	66	0.90	0.03	4
Centennial	ACER	30	99	0	0.87	0.08	1
	POT MCMC	70	93	70	0.90	0.04	24
Millennial	ACER	29	97	0	0.81	0.14	1
	POT MCMC	71	93	70	0.90	0.06	30

From table 4.4, the VaR estimation for the POT MCMC methods seems to consistently outperform the ACER method in accuracy and interval width. The #in CI show signs that the ACER methods 95% confidence interval is overestimated for the i.i.d. Pareto data. Considering the PI, the POT MCMC method also achieve a mean closer to the desired 0.90 exact interval while having lower variation. The #Best CI and #Best PI indicates that the POT MCMC more frequently result in slimmer PI and credible interval than the ACER methods PI and

confidence interval. Noting that the theory of confidence interval differs from credible interval, the comparison is not fully conclusive, but gives an indication of the difference methods estimation variation. As the POT MCMC method is constructed on the i.i.d. assumption, the better performance was expected for the i.i.d. data series. The ACER methods capability of capturing data dependency is not necessary for the generated Pareto distributed data. There are strong evidence suggesting that the POT MCMC method is preferable over the ACER method for i.i.d. observation.

4.1.2 AR(3)-APARCH(1,1) Generated Data

The AR(3)-APARCH(1,1) generated data are clearly dependent and non-stationary. Compared to the generated Pareto distributed points, the data also lack the ability of extracting exact analytical values from the extreme, hence making test and method comparison more challenging. Test verification of the ACER and POT MCMC method for the generated data is achieved by numerically analysing a test set.

The AR(3)-APARCH(1,1) with parameters as described in chapter 3.1.2 is used to generate a data set of size 9 134 125. The data is divided such that the first 9 125 data is used as training set, whereas the remaining 9 125 000 is used as test set. As for the Pareto distribution above, the correspond size could come from having a daily observation for 25 years as training, while a daily observation for 25 000 years as test set. The large test set is for increased evaluation accuracy for the different methods.

The $1-\alpha$ quantile of the test set is used as VaR_α estimation. The amount of the future data falling within the PI is estimated by extracting every nth extreme of the test set, for a chosen n, and calculating the number within the PI divided by total number of nth extremes. The extracted nth extreme is by the definition in chapter 4.2 and 4.2 chosen as the largest value of every continues interval of length n.

As for the Pareto distributed data, a walk-through analyzing an example is presented for a better understanding of the methods and result interpretation. Starting with the ACER method, k is no longer equal to 1 as the data is dependent. A k-plot is necessary for k selection, see figure 4.5. From the plot, the ACER function for $k = 4$ coincides with the ACER functions for higher k values at the tails, hence $k = 4$ is selected for further analysis. For lower k values, the ACER functions differs noticeably in the region for yearly maximum. The 4th ACER function with confidence lines can be seen in figure 4.6. The point where regu-

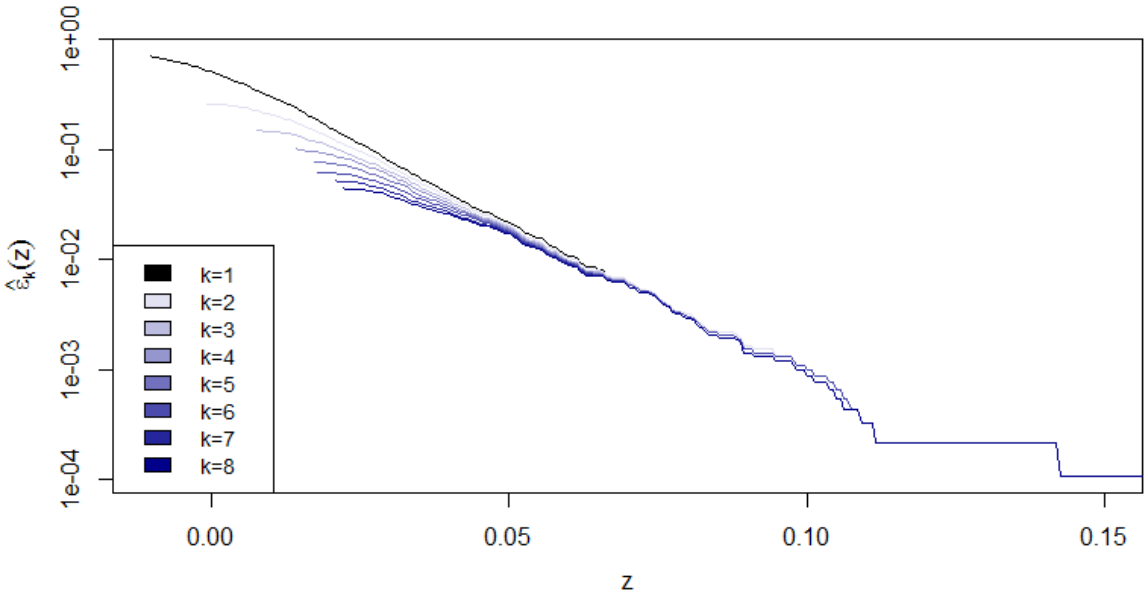

Figure 4.5: The AR(3)-APARCH(1, 1) corresponding k-plot, where the ACER function is plotted against exceedance level for multiple k-dependencies.

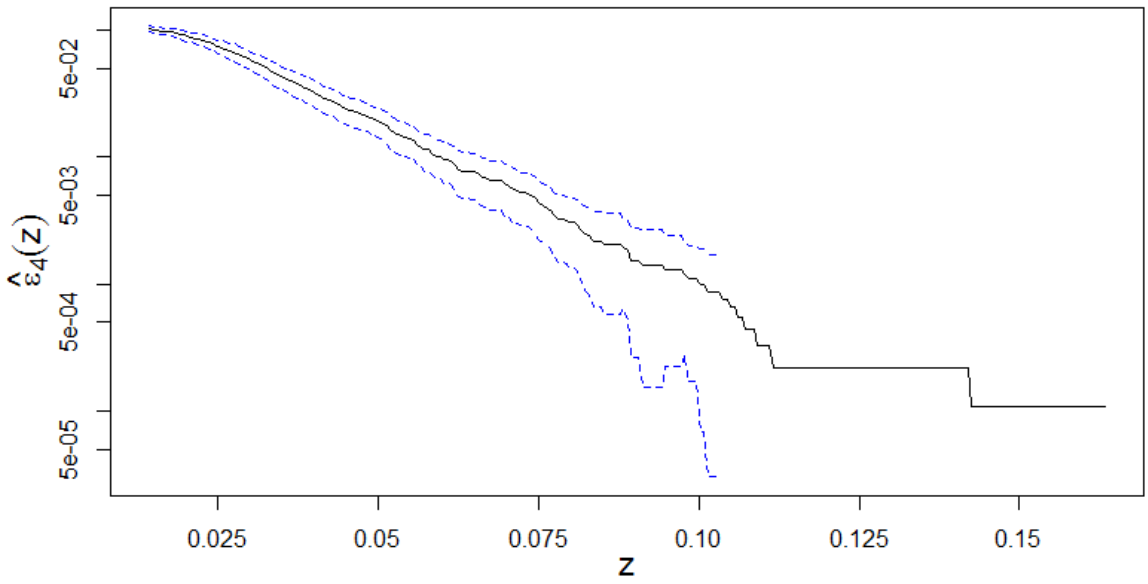

Figure 4.6: The empirical ACER function with 95% confidence interval lines, for $k = 4$. Solid black line indicate the empirical ACER function, while the dashed blue line is the corresponding confidence lines.

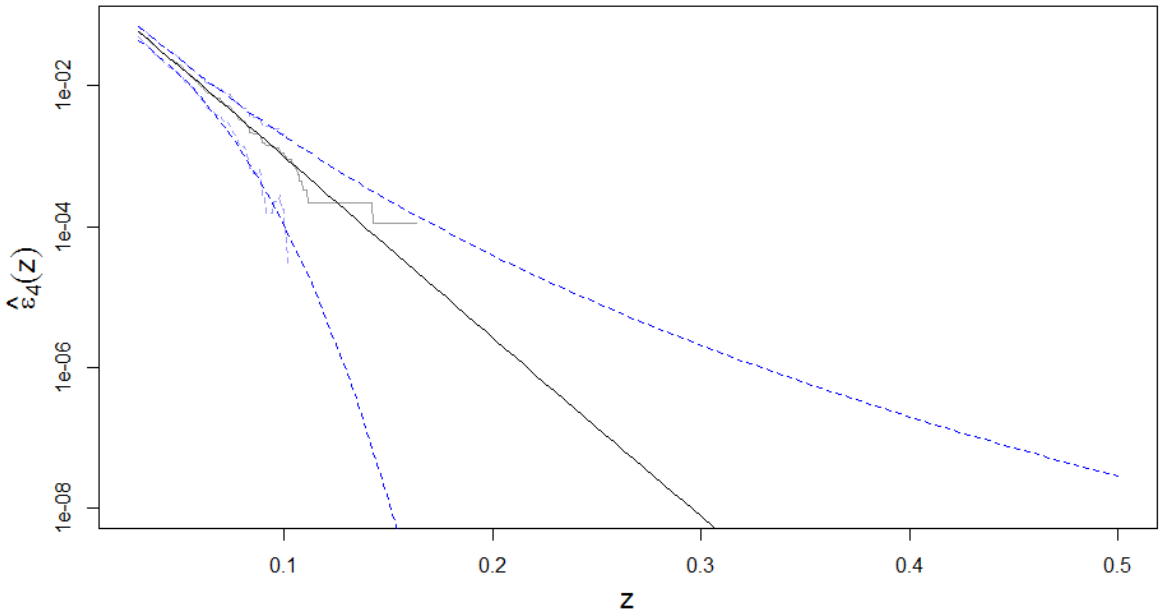

Figure 4.7: The extrapolation of the ACER function shown in solid black line with dashed blue 95% confidence line. A transparent plot of the empirical function as in figure 4.6 is also contained in the plot for estimation validation.

lar tail behavior starts is set to $z_1 = 0.03$, since the bend until $z = 0.025$ shows sign that the regular tail behavior have not settled in yet. Figure 4.7 shows the estimated ACER function with estimated 95% confidence interval lines, together with the empirical plot figure 4.6. The empirical lines fits well with the extrapolation.

Declustering is necessary for the POT MCMC method as the data is dependent. The mean excess plot shown in figure 4.8 was plotted for multiple r values. The behavior of $r = 3$ seems beneficial as the threshold can be selected low, while the point for where the graph change to liner behavior is apparent. The change to linearity can be seen around $u = 0.035$, from where a linear line can be drawn right without crossing the confidence lines, suggesting a threshold at 0.035. Validating the suggested threshold is done through the parameter plot, see figure 4.9. Considering the confidence intervals, both parameters seems constant from 0.035, supporting the suggested threshold, consequently the threshold is set to $u = 0.035$. For the chosen r and u, the POT MCMC method was set to generate 100 000 realizations. In figure 4.10 the first 3 000 realizations of ξ and σ is shown for starting condition set to $[\xi^0, \sigma^0] = [1, 1]$. The series converge within a few hundred iterations. Data between 1st and 500th realization is selected as burn-in and discarded, resulting in an effective sample size of

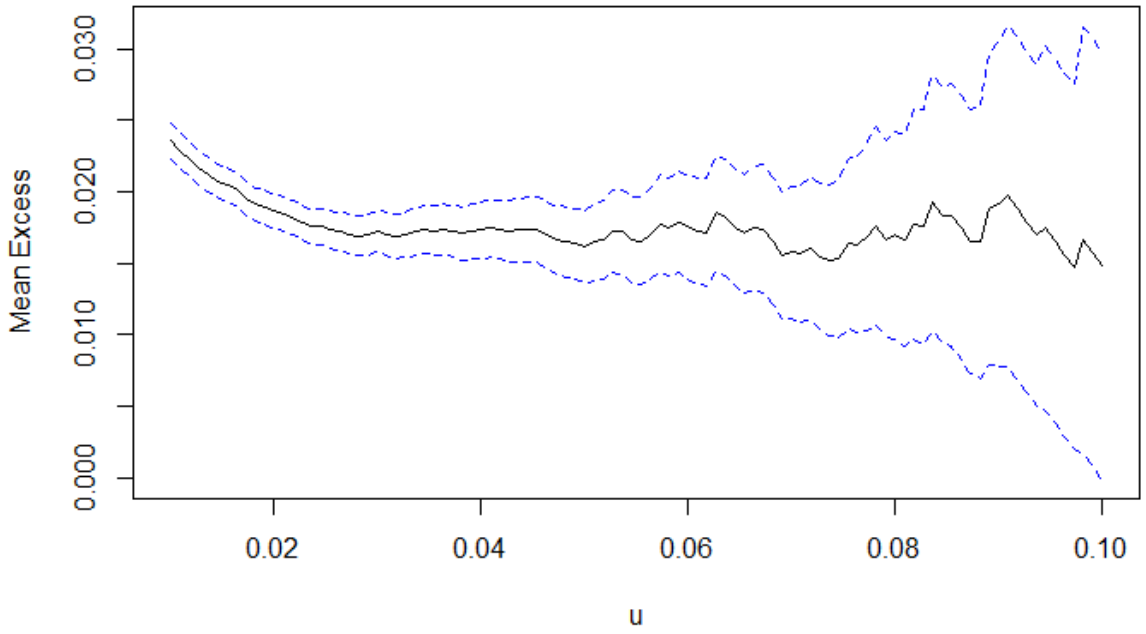

Figure 4.8: Plot of mean excess against threshold u for $r = 3$ in solid line, with 95% confidence interval in dashed blue lines.

13 693 for ξ and 13893 for σ. The POT MCMC method corresponding extrapolation plot can be seen in figure 4.11. The two method achieve similar extrapolation plots, while the POT MCMC credible interval lines are slightly narrower than the ACER confidence interval lines.

For a better understanding of the performance difference, the two methods were applied to 25 AR(3)-APARCH(1,1) generated time series with training and test set as described above. For each training set the predefined values were hold at $z_1 = 0.03$, $k = 4$, $r = 3$ and $u = 0.035$. A summary of the resulting ACER and POT MCMC estimation of the training set is compared in table 4.5. The table layout is equivalent to table 4.4, where the numerically estimated VaR and PI for the test set, is used for judgment instead of the exact values. Referring to the description in chapter 4.1.1, for table interpretation and column names. For estimation of yearly VaR the methods seems to preform close to equally well, while the POT MCMC method is beneficial for more extreme cases. The ACER method 95% confidence interval, captures the numerically estimated VaR an average of 81% of the times, while 66% of the times for the POT MCMC method. Either the variance of the POT MCMC method is heavily underestimated for the AR(3)-APARCH(1,1) dependent data, or the variance of the numerical estimated VaR for the test set is not negligible although the test set is quite large. Future

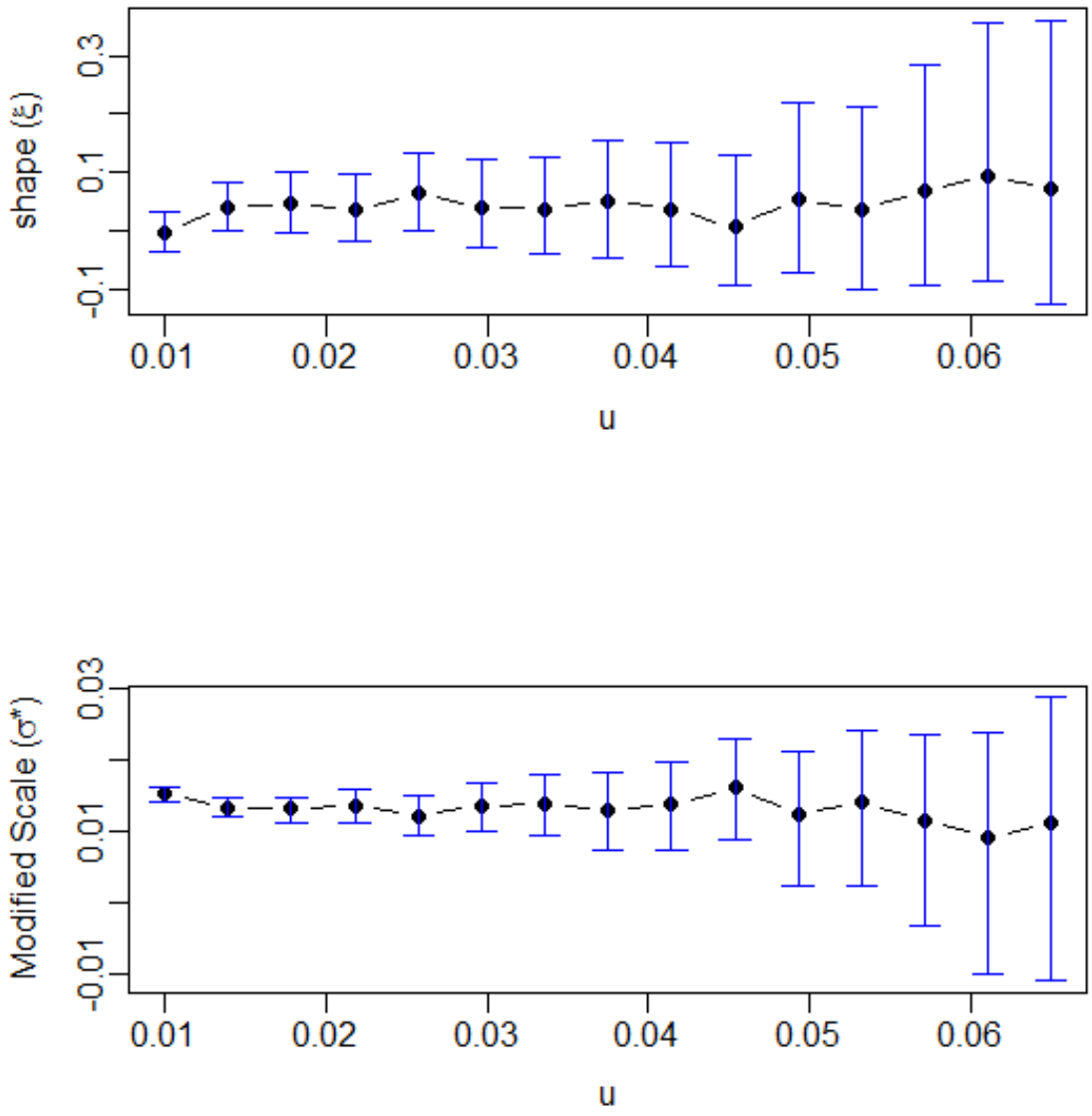

Figure 4.9: Plot of estimated shape and modified scale parameters against threshold. The corresponding 95% credible interval are included in blue.

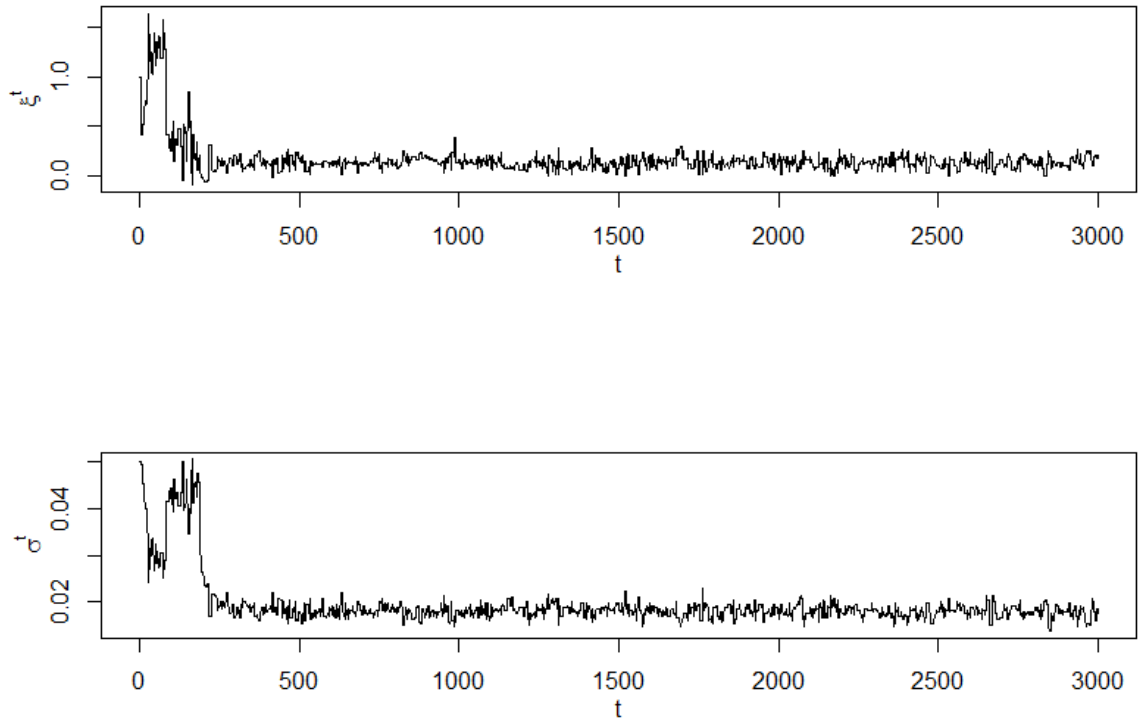

Figure 4.10: The first 3 000 realizations of the POT MCMC method for ξ and σ.

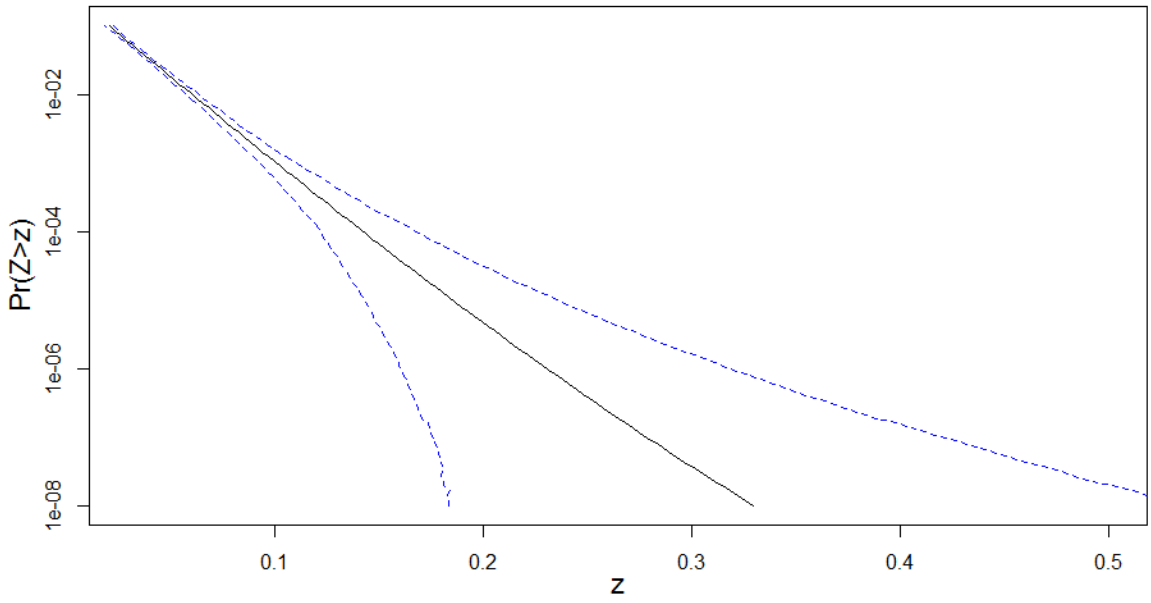

Figure 4.11: The POT MCMC extrapolation plot, with mean in black solid line and credible interval in dashed blue lines.

Table 4.5: A summary of the combined result for the 25 AR(3)-APARCH(1,1) generated time series.

		Estimated VaR			90% Prediction Interval		
		#Best VaR	#in CI	#Best CI	Mean	SD	#Best PI
Yearly	ACER	12	21	0	0.73	0.07	0
	POT MCMC	13	14	13	0.73	0.06	2
Decadal	ACER	6	20	0	0.73	0.12	1
	POT MCMC	19	16	18	0.76	0.12	8
Centennial	ACER	8	20	2	0.61	0.25	4
	POT MCMC	17	19	13	0.80	0.20	10
Millennial	ACER	8	20	2	0.42	0.34	4
	POT MCMC	17	17	12	0.79	0.29	10

analysis can be necessary for better understanding the indication of underestimated confidence and credible interval for dependent time series. The #Best CI becomes invalid as the reliability of the interval width is uncertain. The POT MCMC PI generally captures a higher percentage of the future values than the ACER method, while the variance in performance is lower. The #Best PI also favors the POT MCMC method as more accurate and better tuned for predicting future events.

4.2 Commodity Data

The following section presents the analysis of the observed daily return of the commodity data and the standardized residuals of the corresponding AR(3)-APARCH(1,1) filtered data, presented in chapter 3.2. The Kupiec and Christoffersens test statistics described in 2.5.3 is used for evaluating future VaR estimation, as well as future PI. For a chosen data set, the test statistics is applied by calculating the VaR, and PI using the first 15 years of data, and evaluating them against the upcoming day. The I_1 defined in chapter 2.5.3 is evaluated, and the upcoming day is included in the training set for the next evaluation I_2. The full $\boldsymbol{I}$ vector is calculated by incrementally repeating the process until the test set is empty. The first training set archive a length of 2 818, while there are a total of 1 651 evaluations for each test of the commodity.

The process of manually finding the predefined values of k, z_1, r and u becomes unreasonable slow for each training set of the commodity. Instead the predefined values achieved by the full commodity data series of 4 469 observation is selected, and used for each of the training set for that commodity. The predefined values for each commodity and filtered

commodity can be found in table 4.6. The numbers of POT MCMC iteration is reduced to

Table 4.6: Predefined k and z_1 for the ACER method, with the predefined r and u for the POT MCMC method.

	ACER		POT MCMC	
Data	k	z_1	r	u
CL1	4	0.020	2	0.030
-CL1	3	0.018	2	0.035
HO1	3	0.016	1	0.037
-HO1	2	0.023	1	0.020
NG1	4	0.035	2	0.020
-NG1	4	0.014	1	0.020
CT1	4	0.020	6	0.025
-CT1	4	0.018	2	0.02
C1	5	0.018	3	0.018
-C1	4	0.016	3	0.017
W1	4	0.022	3	0.024
-W1	4	0.020	1	0.030
S1	3	0.015	0	0.014
-S1	4	0.017	1	0.020
RR1	5	0.018	3	0.024
-RR1	4	0.019	2	0.022
SM1	4	0.020	3	0.020
BO1	4	0.016	1	0.020
-BO1	4	0.016	1	0.020
fCL1	1	1.00	0	1.50
fHO1	1	1.50	0	1.80
fNG1	2	1.60	0	1.90
fCT1	2	1.00	0	1.50
fC1	1	1.00	0	1.80
fW1	2	1.00	0	1.40
fS1	1	1.00	0	1.50
fRR1	2	1.00	0	1.25
fSM1	2	1.00	0	1.25
fBO1	1	1.00	0	1.80

32 000 because of the computational intensity. For reliability purposes, the first 2 000 iteration are discarded as burn in, compared to the 500 for the synthetic analysis. The resulting effective sample size for multiple commodity data sets, were close to 4 200. The last parameter realization from the POT MCMC method is passed to the next training set for improved convergence speed. For the daily return commodity data series, both the upper and lower tail are evaluated. The lower tail results are presented with a minus sign in front of the commodity name for the different tables.

For the ACER and POT MCMC method, the Kupiec and Christoffersens test statistics are

applied to the VaR_α for each of the α values 0.05, 0.01, 0.005 and 0.001. Following the definition of nth extreme as in chapter and , the evaluation of the corresponding I_t of the PI is only calculated if the upcoming day is the maximum of the next nth values. Hence the I_t for the PI are commonly separated by a large number of points, making the dependency analysis for the Christoffersens test unnecessary. Consequently only the Kupiec test is applied in the PI evaluation.

Since there are 1 651 test, the expected number of VaR exceedances for a given α is $1651 \cdot \alpha$. In table 4.7, the number of one step out of sample exceedances of the estimated VaR_α for the ACER and POT MCMC method are given for each of the data sets. For table 4.7, 4.8 and 4.9 each cell gives the corresponding ACER and POT MCMC value separated by a backslash (ACER\POT MCMC). In situations where the test favored the ACER method the cell is colored green, and red when the POT MCMC method is favored. The cell is white when the test is rejected on a 5% significance level for both, inconclusive or the methods perform equally well. Table 4.8 present the corresponding p-values of the Kupiec and Christoffersen test statistics for the number of VaR exceedances. Multiple of the 0.05 VaR test where rejected, indicating that both the ACER and POT MCMC methods perform poorly at this level. The 0.05 VaR often fall close to or even below the ACER z_1 and the POT MCMC u for the commodity data series. Hence multiple of the 0.05 VaR are below the lower limits for where the extreme value statistics are preferable compared to other statistical method. Both the POT MCMC and ACER method preformed poorly for the wheat (W1) and corn (C1) data series. Referring to figure 3.3 it is noted that the whole test period for both data sets falls within a higher variance cluster than the majority of the training intervals. This is probably the reason for the poor test result for the two data series. Generally both the ACER and POT MCMC method performed well for the filtered data, where only a few of the test results are rejected on a 5% significant level.

Table 4.9 present the p-value of the Kupiec test statistics for future nth extreme PI interval. The table have some cells where the test statistics were inconclusive represented by NaN. The test becomes inconclusive where no 1 000 days extreme is observed before the last 999 days, as the remaining test set is too short to define the value as a nth extreme.

Without considering dependency for the VaR, the Kupiec test presented in table 4.8 indicate that the ACER method was preferable 10 times compared to the POT MCMC method 12 times for the commodity daily returns, while 9 and 14 times respectively for the filtered data.

Considering the data dependency, the Christoffersen test presented in table 4.8 indicate that the ACER method was preferable 10 times compared to the POT MCMC method 8 times for the commodity daily returns, while 6 and 14 times respectively for the filtered data Considering the Kupiec test for the estimation of PI presented in table 4.9, the ACER method was preferable 10 times compared to the POT MCMC method 12 times for the commodity daily returns, while 1 and 3 times respectively for the filtered data.

Table 4.7: The number of one step out of sample exceedances for the estimated VaR by the ACER\POT MCMC methods.

α	0.05	0.01	0.005	0.001
Expected	83	17	8	2
CL1	109\96	26\23	17\17	11\7
-CL1	100\87	30\30	18\17	2\2
HO1	67\67	15\12	7\7	2\2
-HO1	78\80	13\11	7\6	1\0
NG1	89\90	17\14	9\8	2\2
-NG1	73\63	4\2	1\1	0\0
CT1	172\150	32\27	12\11	1\2
-CT1	153\143	31\28	12\12	2\2
C1	360\272	64\69	35\36	3\3
-C1	230\228	57\61	31\34	8\10
W1	224\223	68\68	36\32	7\8
-W1	218\189	64\61	41\29	13\4
S1	158\132	28\27	15\12	3\2
-S1	159\123	32\30	16\16	3\3
RR1	185\139	5\5	1\1	0\1
-RR1	115\106	16\12	9\9	0\0
SM1	185\178	27\30	14\15	1\1
-SM1	161\147	32\33	16\17	3\4
BO1	145\116	24\23	14\14	4\4
-Bo1	139\129	28\28	14\14	4\4
fCL1	67\71	13\13	7\8	2\2
fHO1	68\70	12\12	8\9	2\2
fNG1	76\69	13\14	7\8	2\2
fCT1	84\79	15\15	3\3	0\0
fC1	95\92	13\14	6\6	1\1
fW1	101\93	26\26	10\12	1\2
fS1	93\95	14\13	9\8	2\2
fRR1	95\78	12\11	7\7	1\1
fSM1	77\78	10\10	3\4	1\2
fBO1	83\85	8\8	5\4	1\1

Table 4.8: The p-values for the number of VaR exceedances using the LR_{UC} and LR_{CC} tests.

	LR_{UC}				LR_{CC}			
Data\α	0.05	0.01	0.005	0.001	0.05	0.01	0.005	0.001
CL1	0.00\0.14	0.03\0.13	0.01\0.01	0.00\0.00	0.00\0.00	0.02\0.04	0.03\0.03	0.00\0.01
-CL1	0.06\0.61	0.00\0.00	0.00\0.00	0.79\0.79	0.00\0.00	0.00\0.00	0.00\0.00	0.97\0.97
HO1	0.07\0.07	0.70\0.24	0.65\0.65	0.79\0.79	0.00\0.00	0.93\0.50	0.90\0.90	0.97\0.97
-HO1	0.60\0.77	0.37\0.15	0.65\0.41	0.58\0.07	0.00\0.00	0.67\0.35	0.90\0.71	0.86\0.19
NG1	0.47\0.41	0.90\0.52	0.80\0.93	0.79\0.79	0.44\0.43	0.38\0.22	0.97\1.00	0.97\0.97
-NG1	0.27\0.02	0.00\0.00	0.00\0.00	0.07\0.07	0.01\0.00	0.00\0.00	0.01\0.01	0.19\0.19
CT1	0.00\0.00	0.00\0.02	0.22\0.36	0.58\0.79	0.00\0.00	0.00\0.00	0.10\0.11	0.86\0.97
-CT1	0.00\0.00	0.00\0.01	0.22\0.22	0.79\0.79	0.00\0.00	0.00\0.00	0.47\0.10	0.97\0.97
C1	0.00\0.00	0.00\0.00	0.00\0.00	0.35\0.35	0.00\0.00	0.00\0.00	0.00\0.00	0.64\0.64
-C1	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00
W1	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00	0.01\0.00
-W1	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.12	0.00\0.00	0.00\0.00	0.00\0.00	0.00\0.00
S1	0.00\0.00	0.01\0.02	0.03\0.22	0.35\0.58	0.00\0.00	0.03\0.04	0.11\0.47	0.64\0.86
-S1	0.00\0.00	0.00\0.00	0.02\0.02	0.35\0.35	0.00\0.00	0.00\0.00	0.02\0.02	0.64\0.64
RR1	0.00\0.00	0.00\0.00	0.00\0.00	0.07\0.58	0.00\0.00	0.00\0.00	0.00\0.00	0.19\0.86
-RR1	0.00\0.01	0.90\0.24	0.80\0.80	0.07\0.07	0.00\0.00	0.99\0.50	0.97\0.97	0.19\0.19
SM1	0.00\0.00	0.02\0.00	0.07\0.03	0.58\0.58	0.00\0.00	0.01\0.00	0.05\0.03	0.86\0.86
-SM1	0.00\0.00	0.00\0.00	0.02\0.01	0.34\0.12	0.00\0.00	0.00\0.00	0.00\0.00	0.64\0.30
BO1	0.00\0.00	0.08\0.13	0.07\0.07	0.12\0.12	0.00\0.00	0.22\0.32	0.19\0.19	0.30\0.30
-BO1	0.00\0.00	0.01\0.01	0.07\0.07	0.12\0.12	0.00\0.00	0.00\0.00	0.00\0.00	0.30\0.30
fCL1	0.07\0.18	0.37\0.37	0.65\0.93	0.79\0.79	0.18\0.39	0.67\0.67	0.97\1.00	0.97\0.97
fHO1	0.09\0.15	0.24\0.24	0.93\0.80	0.79\0.79	0.23\0.33	0.50\0.50	1.00\0.97	0.97\0.97
fNG1	0.45\0.12	0.37\0.52	0.65\0.93	0.79\0.79	0.69\0.28	0.67\0.82	0.90\1.00	0.97\0.97
fCT1	0.87\0.60	0.70\0.70	0.03\0.03	0.07\0.07	0.01\0.00	0.93\0.93	0.11\0.11	0.19\0.19
fC1	0.17\0.29	0.37\0.52	0.41\0.41	0.58\0.58	0.29\0.47	0.67\0.82	0.71\0.71	0.86\0.86
fW1	0.04\0.25	0.03\0.03	0.56\0.22	0.58\0.79	0.09\0.36	0.07\0.07	0.84\0.47	0.86\0.97
fS1	0.25\0.17	0.52\0.37	0.80\0.93	0.79\0.79	0.03\0.02	0.82\0.67	0.97\1.00	0.97\0.97
fRR1	0.17\0.60	0.24\0.15	0.65\0.65	0.58\0.58	0.36\0.82	0.50\0.35	0.90\0.90	0.86\0.86
fSM1	0.53\0.60	0.08\0.08	0.03\0.10	0.58\0.79	0.49\0.50	0.22\0.22	0.11\0.26	0.86\0.97
fBO1	0.96\0.78	0.02\0.02	0.22\0.10	0.58\0.58	0.04\0.04	0.06\0.06	0.47\0.26	0.86\0.86

Table 4.9: The p-value of the Kupiec test statistics for future nth extremes PI interval.

Data\n	100	200	1000
CL1	0.69\0.69	0.04\0.04	0.02\0.43
-CL1	0.74\0.74	0.11\0.11	0.52\0.52
HO1	0.02\0.02	0.19\0.82	0.65\0.65
-HO1	0.43\0.94	0.14\0.14	0.52\0.52
NG1	0.94\0.54	0.06\0.06	NaN\NaN
-NG1	0.18\0.18	0.62\0.62	0.65\0.65
CT1	0.10\0.77	0.17\0.17	0.52\0.52
-CT1	0.28\0.31	1.00\1.00	NaN\NaN
C1	0.38\0.38	0.06\0.91	NaN\NaN
-C1	0.01\0.01	0.82\0.04	NaN\NaN
W1	0.68\0.07	0.35\1.00	NaN\NaN
-W1	0.00\0.07	0.00\0.72	NaN\NaN
S1	0.65\0.65	0.82\0.82	0.03\0.03
-S1	0.38\0.13	0.22\0.04	NaN\NaN
RR1	0.45\0.43	1.00\1.00	0.65\0.65
-RR1	0.60\0.90	0.09\0.71	0.43\0.43
SM1	0.41\0.41	0.84\0.84	0.36\0.36
-SM1	0.65\0.68	0.22\0.17	NaN\NaN
BO1	0.43\0.15	0.02\0.02	0.65\0.03
BO1	0.61\0.05	0.12\0.12	0.52\0.52
fCL1	0.81\0.81	0.22\0.22	0.64\0.64
fHO1	0.61\0.61	0.36\0.36	0.65\0.65
fNG1	0.81\0.81	0.22\0.22	0.65\0.65
fCT1	0.09\0.09	0.17\0.17	NaN\NaN
fC1	0.64\0.64	0.82\0.82	NaN\NaN
fW1	0.77\0.77	0.19\0.19	NaN\NaN
fS1	0.75\0.75	0.51\0.51	0.03\0.03
fRR1	0.59\0.59	0.22\0.22	0.27\0.43
fSM1	0.50\0.88	0.22\0.72	0.65\0.65
fBO1	0.94\0.94	0.28\0.28	0.65\0.03

Chapter 5

Conclusion

Through this work, an AMCMC method for the POT extreme value statistics have been developed. The ACER method was evaluated against the POT MCMC method through synthetically generated points, observed commodity daily returns and filter commodity daily returns. The methods were tested for the accuracy of the extrapolated VaR as well as the PI for the maximum of n future observations.

Generally, the POT MCMC method seems beneficial over the ACER method for forecasting PI. For the synthetic i.i.d. Pareto distributed data, the POT MCMC method performed significantly better with respect to VaR estimation and forecasting of nth extreme PI. For the generated AR(3)- APARCH$(1,1)$ skewed student t-distributed data the ACER VaR extrapolation and PI forecasting fall short compared to POT MCMC method, while possible performing better with respect to the extrapolated VaR confidence interval. The POT MCMC method seems to perform better for the AR(3)- APARCH$(1,1)$ filtered commodity data with respect to VaR and PI. Since none of the filtered commodity data was rejected by the Ljung-Box test statistics with respect to i.i.d., the filtered data has close to i.i.d. behavior. Hench suggesting that the POT MCMC method is superior over the ACER method for i.i.d. observation, or close to i.i.d. While not conclusive, there are signs that the extrapolated VaR for the ACER method might captures the VaR dependencies better than the POT MCMC method for the nonfiltered commodity daily returns.

5.1 Recommendations for Further Work

Future analysis within the field is needed for more conclusive results. Analysis using additional commodity data will reduce the variability in the overall result, such that more definitive conclusions can be made. Future work for constructing an approach that captures the parameter variability of the ACER method could potentially improve the extrapolated confidence interval as well as the PI. A deeper investigation of the POT prior distribution for the MCMC method can probably improve the POT MCMC method, as the priors for this work was unnecessary wide.

Appendix A

Acronyms

i.i.d. independent and identically distributed

GEV Generalized Extreme Value

GPD Generalized Pareto Distribution

POT Peak Over Threshold

ACER Average Conditional Exceedance Rate

MCMC Markov Chain Monte Carlo Method

AMCMC Adaptive Markov Chain Monte Carlo Method

VaR Value at Risk

PI Prediction Interval

AR Autoregressive

MA Moving Average

ARMA Autoregressive Moving Average

ARCH Autoregressive Conditional Heteroscedasticity

GARCH Generalized Autoregressive Conditional Heteroscedasticity

APARCH Asymmetric Power Autoregressive Conditional Heteroscedasticity

GJR Glosten Jagannathan Runkle

Appendix B

C++ and R Code

The appendix contains the developed C++ and R codes necessary for this work.

B.1 POT MCMC

Object-oriented programing was used for the mcmc.gdp program, such that the function could be applied to raw data as well as adding realizations to an existing MCMC class.

The internal C++ code, for increased speed.

```
#include <Rcpp.h>
#include <ctime>
#include <iostream>
#include <vector>
#include <math.h>
#include <float.h>
#include <limits>
using namespace Rcpp;

const double epsilon = std::numeric_limits<double>::min();
const double two_pi = 2.0*3.14159265358979323846;
const double CPPMIN= -std::numeric_limits<float>::infinity();

double * mvrnormC(double mu0, double mu1, double sigma[4]) {
  static double mrn[2];
  double r[2];
```

```
17   double l[3];
18   // Cholesky decomposition sigma=l%%t(l)
19   l[0] = sqrt(sigma[0]);
20   l[1] = sigma[1] / l[0];
21   l[2] = sqrt(sigma[3] - l[1] * l[1]);
22
23   // u1,u2 ~ runif [0,1]
24   double u1, u2;
25   do
26   {
27     u1 = rand()*(1.0 / RAND_MAX);
28     u2 = rand()*(1.0 / RAND_MAX);
29   } while (u1 <= epsilon);
30   // Box Muller transform r1,r2 proportionalt to rnorm01
31   r[0] = sqrt(-2.0 * log(u1)) * cos(two_pi * u2);
32   r[1] = sqrt(-2.0 * log(u1)) * sin(two_pi * u2);
33   // mrn=mu+l%*%r
34   mrn[0] = mu0 + l[0] * r[0];
35   mrn[1] = mu1 + l[1] * r[0] + l[2] * r[1];
36   return(mrn);
37 }
38
39 //should work
40 double lnRGdp(double *y, int yn,double ymax, double Xtemp1, double
       Xtemp2, double X1, double X2) {
41   if (Xtemp1 < (-exp(Xtemp2) / ymax)) {
42     return(CPPMIN);
43   }
44   double logLikeXtemp = 0;
45   double logLikeX = 0;
46   double invExpXtemp2 = exp(-Xtemp2);
47   double invExpX2 = exp(-X2);
48   //logNormalPdf=log(f(Xtemp1)*f(Xtemp2)/(f(X1)*f(X2))), where f is
       normal
```

```
49  //var(x1)=100, var(x2)=1000 =>0.01 and 0.0001, mu(x1)=mu(x2)=0
50  double logNormalPdf = -0.5*((Xtemp1 - X1)*(Xtemp1 + X1)*0.01 + (
     Xtemp2 - X2)*(Xtemp2 + X2)*0.0001);
51  // xi=0:  l(sigma)=-k*log(sigma)-sigma^(-1)*sum(yi) p80, Stuart
     Coles, An Introduction...
52  // xi!=0: l(sigma)=-k*log(sigma)-(1+1/xi)*sum(log(1+xi*yi/sigma))
      p80, Stuart Coles, An Introduction...
53  if (Xtemp1 == 0) {
54    for (int i = 0; i < yn; i++) {
55      logLikeXtemp += y[i];
56    }
57    logLikeXtemp = -logLikeXtemp * invExpXtemp2;
58    logLikeXtemp -= yn*Xtemp2;
59  }
60  else {
61    for (int i = 0; i < yn; i++) {
62      logLikeXtemp += log(1 + Xtemp1*y[i] * invExpXtemp2);
63    }
64    logLikeXtemp = -logLikeXtemp * (1 + 1 / Xtemp1);
65    logLikeXtemp -= yn*Xtemp2;
66  }
67  if (X1 == 0) {
68    for (int i = 0; i < yn; i++) {
69      logLikeX += y[i];
70    }
71    logLikeX = -logLikeX * invExpX2;
72    logLikeX -= yn*X2;
73  }
74  else {
75    for (int i = 0; i < yn; i++) {
76      logLikeX += log(1 + X1*y[i] * invExpX2);
77    }
78    logLikeX = -logLikeX * (1 + 1 / X1);
79    logLikeX -= yn*X2;
```

```
80    }
81    return(logLikeXtemp - logLikeX + logNormalPdf);
82  }
83
84  // [[Rcpp::export]]
85  Rcpp::List mcmcGpdC(NumericVector y, int nstart, int n,
        NumericVector start, NumericVector muR, NumericMatrix varR,
        double tau, double a, double gam, double lamda)
86  {
87    srand((unsigned)time(NULL));
88    //double total1, total2;
89    double mu[2], var[4];
90    double lamdaxVar[4];
91    double R;
92    double gamC = gam;
93    double uni;
94    double *Xtemp;
95    double ymax= -DBL_MAX;
96    int yn = y.size();
97    double yC[yn];
98    //vector (y) to array (yC)
99    std::copy(y.begin(), y.end(), yC);
100   std::copy(muR.begin(), muR.end(), mu);
101   std::copy(varR.begin(), varR.end(), var);
102   //double *mrn;
103   std::vector<double> thetaXi(n), thetaPhi(n);
104   thetaXi[0]= start[0], thetaPhi[0] = start[1];
105   //find largest y (ymax)
106   for (int j = 0; j < yn; j++) {
107     if (ymax < yC[j]) {
108       ymax = yC[j];
109     }
110   }
111   if (lamda == 0) {
```

```
112     lamda = pow(2.38, 2.0) / 2;
113   }
114   for (int i = 1; i <n; i++) {
115     if (tau != 0) {
116       gamC = 0.5*exp(-(i + nstart) / tau);
117     }
118     for (int k = 0; k < 4; k++) lamdaxVar[k] = var[k] * lamda;
119     Xtemp = mvrnormC(thetaXi[i - 1], thetaPhi[i - 1], lamdaxVar);
120
121     R = lnRGdp(yC, yn, ymax, *(Xtemp), *(Xtemp + 1), thetaXi[i -
    1], thetaPhi[i - 1]);
122     //R = -0.3566;
123     uni = log(rand()*(1.0 / RAND_MAX));
124     if (uni < R) {
125       thetaXi[i] = *(Xtemp);
126       thetaPhi[i] = *(Xtemp+1);
127     }
128     else {
129       thetaXi[i] = thetaXi[i-1];
130       thetaPhi[i] = thetaPhi[i-1];
131     }
132     if (gamC > DBL_EPSILON) {
133       if (a != 0) {
134         if (R >= 0) {
135           R = 1;
136         }
137         else {
138           R = exp(R);
139         }
140         lamda = lamda*exp(gamC*(R - a));
141       }
142       // Matrix calculation: var=var+gam*((theta[,i]-mu)%*%t(theta
    [,i]-mu)-var)
143       var[0] = var[0] + gamC*((thetaXi[i] - mu[0])*(thetaXi[i] - mu
```

```
    [0]) -var[0]);
144         var[1] = var[1] + gamC*((thetaXi[i] - mu[0])*(thetaPhi[i] -
    mu[1]) - var[1]);
145         var[3] = var[3] + gamC*((thetaPhi[i] - mu[1])*(thetaPhi[i] -
    mu[1]) - var[3]);
146         var[2] = var[1];
147         // Matrix calculation: mu<-mu+gam*(theta[,i]-mu)
148         mu[0] = mu[0] + gamC*(thetaXi[i] - mu[0]);
149         mu[1] = mu[1] + gamC*(thetaPhi[i] - mu[1]);
150       }
151     }
152     std::vector<double> muC(mu, mu + 2);
153     std::vector<double> varC(var, var + 4);
154     return Rcpp::List::create(Rcpp::Named("data")= NAN, Rcpp::Named("
      data") = NAN, Rcpp::Named("theta") = NAN,
155       Rcpp::Named("thetaXi") = thetaXi, Rcpp::Named("thetaPhi") =
    thetaPhi, Rcpp::Named("mu") = muC,
156       Rcpp::Named("var") = varC, Rcpp::Named("n") = n, Rcpp::Named("
    burnin") = NAN, Rcpp::Named("aRate") = NAN,
157       Rcpp::Named("MLE") = NAN, Rcpp::Named("MLEest") = NAN, Rcpp::
    Named("tau") = tau, Rcpp::Named("a") = a,
158       Rcpp::Named("gam") = gam, Rcpp::Named("lamda") = lamda);
159 }
```

Main and internal R functions.

```
1 # Include the C++ function
2 library(Rcpp)
3 sourceCpp("Cpp/MCMC_GPD_CPP.cpp")
4
5 # data is matrix/vector of data (X), u aka thershold, start is
    starting values ([type,number])
6 # type is xi and sigma (=exp(phi)), start=c(xi,sigma) or matrix
    which xi and sigma are row
7 # mu and var is mean and variance for the random walk starting
    distribution (g(x*|x)),
```

```
# gam (gamma) is the adaptive paramter (exp(-x/t)),
# a is the optimal accaptance rate.
# tau=c(gam size, at what n) default tau=c(1/20,0.1*n)

# For numerical input
mcmc.gpd<- function(X, ...) UseMethod("mcmc.gpd",X)
mcmc.gpd.numeric<-function(X,u,n=1000,start=NULL,r=0,mu=NULL,var=
    NULL,a=NULL,gam=NULL,cpp=TRUE,...){
  require(MASS)

  if(is.matrix(X)){
    X=as.vector(t(X))
    X=X[!is.na(X)]
  }
  ny<-length(X)
  y<-ufilt(X=X,u=u,r=r)
  k=length(y)

  #old method without filt
  #y<-X-u
  #ny<-length(y)
  #k<-sum(y>0)
  #y=y[y>0]

  # Fix starting values
  if(is.null(start)){
    start<-matrix(rep(NA,10),2)
    start[,1]<-rep(.Machine$double.eps*100,2)#c(0,1)
    start[,2]<-c(1.5,0.3)
    start[,3]<-c(0.5,3)
    start[,4]<-c(-0.1,0.3)
    start[,5]<-c(-0.5,3)
    # c(xi,sigma), in "Choose starting value" sigma->log(sigma)
  }else if(all(start==0)){
```

```
    start<-c(.Machine$double.eps*100,log(.Machine$double.eps*100))
  }else{
    if(length(start)==2){
      if(start[2]≤0){stop('sigma must be > 0')}
      start<-c(start[1],log(start[2]))
    }else if(is.matrix(start)){
      if(any(start[2,])<0){stop('sigma must be > 0')}
    }else{
      stop('start must be vector or matrix')
    }
  }
  # here start=c(xi, phi) where phi=log(sigma)

  # Starting mu and var for MCMC
  if(is.null(mu)){mu=c(0,0)}
  if(is.null(var)){var<-matrix(c(1,0,0,1),2,2)}

  # Choose starting value
  if(length(start)>2){
    startBest<-c(0,0)
    lnBest<--Inf
    temp<-NA
    for(i in 1:length(start[1,])){
      if(start[1,i]< (-start[2,i]/max(y))){
        start[1,i]<- 0.9*(-start[2,i]/max(y))
      }
      temp<-mcmc.gpd(X=X,u=u,start=start[,i],mu=mu,var=var,n=500,
   gam=0.1,cpp=cpp)
      lnTemp<-lnGpd(y,temp$theta[c(1:2),500])
      if(lnTemp>lnBest){
        lnBest<-lnTemp
        startBest<-c(temp$theta[1,500],log(temp$theta[2,500]))
        mu<-temp$mu
        var<-temp$var
```

```
74        }
75      }
76      start<-startBest
77    }
78    ####################MCMC#########################
79    lamda<-2.38^2/2
80    gamInd<-gam
81    if(is.null(gam)){
82      tau<-0.1*n/log(10)
83      gam=0
84    }else{
85      tau<-0
86    }
87    if(is.null(a)){a=0}
88
89    if(cpp==TRUE){
90      MCMC<-mcmcGpdC(y=y,nstart=1,n=n,start=start,muR=mu,varR=var,
      tau=tau,a=a, gam=gam,lamda=lamda)
91      MCMC$theta<-rbind(MCMC$thetaXi,exp(MCMC$thetaPhi))
92      MCMC$thetaXi<-NULL
93      MCMC$thetaPhi<-NULL
94      MCMC$var<-matrix(MCMC$var,2,2)
95    }else{
96      MCMC<-mcmcGpd.internal(y=y,nstart=1,n=n,start=start,mu=mu,var=
      var,tau=tau,a=a,gam=gam,lamda=lamda)
97    }
98    ################################
99    MCMC$theta<-rbind(MCMC$theta,rbeta(n,k+1,ny-k+1))
100   MCMC$data<-X
101   MCMC$u<-u
102   class(MCMC)<-'mcmc'
103   return(MCMC)
104 }
105
```

```
106 # Adding point to existing mcmc. Has the mcmc object as input
107 mcmc.gpd.mcmc<-function(X,n=1000,r=0,a=NULL,gam=NULL,cpp=TRUE,...){
108   require(MASS)
109   if(!is.null(gam)){X$gam<-gam}
110   if(!is.null(a)){X$a<-a}
111
112   ny<-length(X$data)
113   y<-ufilt(X=X$data,u=X$u,r=r)
114   k=length(y)
115   ##################MCMC###############
116   var<-X$var
117   mu<-X$mu
118   lamda<-X$lamda
119   a<-X$a
120   gam<-X$gam
121   tau<-X$tau
122   start<-c(X$theta[1,X$n],log(X$theta[2,X$n]))
123   if(cpp==TRUE){
124     MCMC<-mcmcGpdC(y=y,nstart=X$n,n=(n+1),start=start,muR=mu,varR=
      var, tau=tau,a=a, gam=gam,lamda=lamda)
125     MCMC$theta<-rbind(MCMC$thetaXi,exp(MCMC$thetaPhi))
126     MCMC$thetaXi<-NULL
127     MCMC$thetaPhi<-NULL
128     MCMC$var<-matrix(MCMC$var,2,2)
129   }else{
130     MCMC<-mcmcGpd.internal(y=y,nstart=X$n,n=(n+1),start=start,mu=mu
      ,var=var,tau=tau,a=a,gam=gam,lamda=lamda)
131   }
132   ###################################
133   X$theta<-cbind(X$theta,rbind(MCMC$theta[c(1,2),2:MCMC$n],rbeta(
      MCMC$n-1,k+1,(ny-k+1))))
134   X$var<-MCMC$var
135   X$mu<-MCMC$mu
136   X$n<-X$n+n
```

```
137   X$lamda<-MCMC$lamda
138   return(X)
139 }
140 
141 # Internal R mcmc program. An alternative to the C++ function, but
      slower.
142 mcmcGpd.internal<-function(y,nstart,n,start,mu,var,tau,a,gam,lamda)
      {
143   if(lamda==0){lamda<-2.38^2/2}
144   theta<-matrix(rep(NA,2*n),2)
145   theta[,1]<-start
146   uni<-log(runif(n-1))
147   R<-NA
148   Xtemp<-NA
149   gamInd<-gam
150   if(tau!=0){
151     gam<-0.5*exp(-(nstart:(nstart+n-1))/tau)
152   }else{
153     gam<-rep(gam,n)
154   }
155   for(i in 2:n){
156     Xtemp<-mvrnorm(n=1,theta[,(i-1)],lamda*var)
157     R=lnRGdp(y,Xtemp,theta[,(i-1)])
158     if(uni[i-1]<R){
159       theta[,i]<-Xtemp
160     }else{
161       theta[,i]<-theta[,(i-1)]
162     }
163 
164     if(gam[i]>.Machine$double.eps){
165       if(a!=0){
166         if(R>=0){R=1
167         }else{R=exp(R)}
168         lamda<-lamda*exp(gam[i]*(R-a))
```

```
169       }
170       var<-var+gam[i]*((theta[,i]-mu)%*%t(theta[,i]-mu)-var)
171       mu<-mu+gam[i]*(theta[,i]-mu)
172     }
173   }
174   theta<-rbind(theta[1,],exp(theta[2,]))
175   MCMC<-list(data=NA,u=NA,theta=theta, mu=mu, var=var, n=n, burnin=
       NA,aRate=NA, MLE=NA,
176               MLEest=NA,tau=tau, a=a, gam=gamInd,lamda=lamda)
177
178   return(MCMC)
179 }
180
181 # l(xi,si)=n*phi-(1+1/xi)*sum(log(1+xi*y/exp(phi))) # phi=log(si)
182 lnRGdp<-function(data,Xtemp,X){
183   dataLength<-length(data)
184   if(Xtemp[1]< (-exp(Xtemp[2])/max(data))){return(-Inf)}
185   lnGPNtemp<--dataLength*Xtemp[2]-(1+1/Xtemp[1])*sum(log(1+Xtemp[1]
       *data/exp(Xtemp[2])))
186   lnpxitemp<-dnorm(Xtemp[1], mean = 0, sd = sqrt(100), log = TRUE)
187   lnpphitemp<-dnorm(Xtemp[2], mean = 0, sd = sqrt(10000), log =
       TRUE)
188   lnGPN<--dataLength*X[2]-(1+1/X[1])*sum(log(1+X[1]*data/exp(X[2]))
       )
189   lnpxi<-dnorm(X[1], mean = 0, sd = sqrt(100), log = TRUE)
190   lnpphi<-dnorm(X[2], mean = 0, sd = sqrt(10000), log = TRUE)
191   return(lnGPNtemp+lnpxitemp+lnpphitemp-lnGPN-lnpxi-lnpphi)
192 }
193
194 # X is parameters (aka theta) matrix of thetas or single theta(c(xi
       ,sigma)), data is a vector.
195 lnGpd<-function(data,X){
196   if(X[1]<(-X[2]/max(data))){return(-Inf)}
197   if(is.matrix(X)){
```

```
198     temp<-rep(NA,length(X[1,]))
199     for(i in 1:length(X[1,])){
200       temp[i]<-sum(log(1+X[1,i]*data/X[2,i]))
201     }
202     return( -length(data)*log(X[2,])-(1+1/X[1,])*temp)
203   }else if(is.numeric(X)){
204     return(-length(data)*log(X[2])-(1+1/X[1])*sum(log(1+X[1]*data/X
      [2])))
205   }
206 }
```

B.2 Effective Sample Size and Acceptance rate

The functions for calculating the Effective Sample Size and acceptance rate.

```
1  # Effective sample size
2  # class(X)= sim of xi, sigma or alpha after burnin
3  effsampSize<-function(X){
4    p<-acf(X,plot=FALSE)[[1]]
5    pleng<-which(p<0.1)[1]
6    if(pleng==2){
7      return(length(X))
8    }else{
9      return(length(X)/(sum(p[2:pleng])*2+1))
10   }
11 }
12
13 #acceptance (a) rate
14 aRate<-function(X){
15   a<-sum(diff(X)!=0)
16   return(a/length(X))
17 }
```

B.3 Mean Excess and Parameter plot

The mean excess (uplot) and parameter (paraplot) plot with internal functions.

```
# plot mean exceedance vs u (thresholds) with confidence lines.
# For valid u's, need linearity.
# X=data set
# r=numbers of data below threshold, before defined as new cluster.
# k=plot ends when only k extreme is left
# iid Independent and Identical Distributed => r=0
# numPoint= number of points used in plot
uplot<-function(X,k=3,r=1,xlim=NULL, CI=0.95,numPoint=100,...){

  if(CI>1||CI<0){stop("CI should be 0 < CI < 1")}
  if(is.matrix(X)){
    X=as.vector(t(X))
    X=X[!is.na(X)]
  }
  if(!(is.vector(xlim)&&length(xlim)==2)){
    Xsort<-sort(X,na.last = NA)
    if(is.null(xlim)){
      xlim=c(Xsort[floor(length(Xsort)*0.75)],NA)
    }else{
      xlim<-c(xlim,NA)
    }
    if(r==0){
      xlim[2]<-Xsort[length(Xsort)-k+1]
    }else{
      xlim[2]<-umax(X,k=k,r=r)
    }
  }
  uStep<-seq(from=xlim[1],to=xlim[2], length=numPoint)
  ymean<-rep(NA,numPoint)
  tSD<-rep(NA,numPoint)
  #the variable is tSD=t*sd
```

```
  lowCI<-rep(NA,numPoint)
  pb <- txtProgressBar(min = 0, max = numPoint, style = 3)
  for(i in 1:numPoint){
    ytemp<-ufilt(X,uStep[i],r=r)
    ymean[i]=mean(ytemp)
    tSD[i]=qt(p=(1-CI)/2, df=length(ytemp)-1, lower.tail = FALSE)*
     sd(ytemp)/sqrt(length(ytemp))
    setTxtProgressBar(pb, i)
  }
  close(pb)
  plot(uStep,ymean,type='l',xlim=xlim,ylim=c(min(ymean-tSD),max(
     ymean+tSD)),xlab = "u", ylab = "Mean Excess")
  lines(uStep, ymean+tSD, col='blue', lty=2)
  lines(uStep, ymean-tSD, col='blue', lty=2)
}

# parameter plot (xi vs u and sigma*=sigma-xi*u \sim const vs u)
# maxtime= maximum time calculating at each u (by defualt 5 seconds
    )
# numPoint= number of points used in plot
parplot<-function(X,k=3,r=1,xlim=NULL,CI=0.95,maxtime=5,burnin=500,
    numPoint=15,...){
  if(CI>1||CI<0){stop("CI should be 0 < CI < 1")}
  if(is.matrix(X)){
    X=as.vector(t(X))
    X=X[!is.na(X)]
  }
  if(!(is.vector(xlim)&&length(xlim)==2)){
    Xsort<-sort(X,na.last = NA)
    if(is.null(xlim)){
      xlim=c(Xsort[floor(length(Xsort)*0.75)],NA)
    }else{
      xlim<-c(xlim,NA)
    }
```

```
62    if(r==0){
63      xlim[2]<-Xsort[length(Xsort)-k+1]
64    }else{
65      xlim[2]<-umax(X,k=k,r=r)
66    }
67  }
68
69  names<-c(expression(paste("shape (", xi,")")), expression(paste("
     Modified Scale (", sigma,"*)")))
70  uStep<-seq(from=xlim[2],to=xlim[1], length=numPoint)
71  para<-matrix(NA,2,numPoint) # para[1,]=xi while para[2,]=sigma*
72  upCI<-matrix(NA,2,numPoint)
73  lowCI<-matrix(NA,2,numPoint)
74
75
76  tempmcmc<-NA
77  tempstart<-rep(NA,2)
78  tempdata<-matrix(NA,2,9000)
79  burnintemp=burnin
80
81  pb <- txtProgressBar(min = 0, max = numPoint, style = 3)
82  for(j in 1:numPoint){
83    if(j==1){
84      tempmcmc<-mcmc.gpd(X,u=uStep[j],n=10000)
85    }else{
86      if((sum(X>uStep[j])>(maxtime*1200))&&(sum(X>uStep[j-1])>30)){
87        timescale<-maxtime*1200/(sum(X>uStep[j]))
88        tempmcmc<-mcmc.gpd(X=X,u=uStep[j],start=tempstart,n=floor(
     timescale*10000),mu=tempmcmc$mu,var=tempmcmc$var)
89        burnintemp=floor(0.2*(timescale*10000))
90      }else if(sum(X>uStep[j-1])>30){
91        tempmcmc<-mcmc.gpd(X=X,u=uStep[j],start=tempstart,n=10000,
     mu=tempmcmc$mu,var=tempmcmc$var)
92        burnintemp=burnin
```

```
93        }else{
94          tempmcmc<-mcmc.gpd(X=X,u=uStep[j],n=10000)
95          burnintemp=burnin
96        }
97      }
98
99      tempstart<-tempmcmc$theta[1:2,tempmcmc$n]
100     tempsort<-rbind(sort(tempmcmc$theta[1,burnintemp:tempmcmc$n]),
      sort(tempmcmc$theta[2,burnintemp:tempmcmc$n]-tempmcmc$theta[1,
      burnintemp:tempmcmc$n]*uStep[j]))
101
102     para[1:2,j]<-c(mean(tempsort[1,]),mean(tempsort[2,]))
103     upCI[1:2,j]<-c(tempsort[1,floor((tempmcmc$n-burnintemp)*(1+CI)/
      2)],tempsort[2,floor((tempmcmc$n-burnintemp)*(1+CI)/2)])
104     lowCI[1:2,j]<-c(tempsort[1,floor((tempmcmc$n-burnintemp)*(1-CI)
      /2)],tempsort[2,floor((tempmcmc$n-burnintemp)*(1-CI)/2)])
105     setTxtProgressBar(pb, j)
106   }
107   close(pb)
108   oldpar <- par(mfrow = c(2, 1))
109   for(i in 1:2){
110     plot(uStep,para[i,], xlab='u', ylab=names[i],ylim=c(min(lowCI[i
      ,]),max(upCI[i,])),pch=16,type='b',mgp=c(2,0.5,0))
111     arrows(uStep,lowCI[i,],uStep,upCI[i,],code=3,length=0.1,angle=9
      0,col='blue')
112   }
113   par(oldpar)
114 }
115
116
117 # Filtering out threshold exceedance y, which is the max of each
      cluster exceeding the treshold.
118 # Cluster is define where r value is below the threshold (
      repeatedly)
```

```
# x=data set (vector)
# u=Threshold
# r=numbers of data below, before defined as new cluster
# y=return x-u
ufilt<-function(X,u,r=1){
  if(is.matrix(X)){
    X=as.vector(t(X))
    X=X[!is.na(X)]
  }
  if(r==0){
    y<-X[X>=u]-u
  }else{
    y<-rep(NA,sum(X>=u))
    max<-u*(1-2*.Machine$double.eps)
    rcount<-0
    clust<-FALSE
    j<-1
    for(i in 1:length(X)){
      if(X[i]>=u){
        rcount<-0
        clust<-TRUE
        if(X[i]>max){
          max<-X[i]
        }
      }else if(X[i]<u&&clust){
        rcount<-rcount+1
      }
      if(rcount>=r&&max>=u){
        y[j]<-max
        clust<-FALSE
        rcount<-0
        max<-u*(1-2*.Machine$double.eps)
        j<-j+1
      }
```

```
    }
    if(max>=u){
      y[j]<-max
    }
    y=y[!is.na(y)]-u
  }
  return(y)
}

# find u-max which gives only
# k block exceedance
# r=numbers of data below threshold, before defined as new cluster.
# return u-max
umax<-function(X,k,r){
  Xmean<-mean(X)
  uLeng<-0
  Xsort<-sort(X)
  n <- length(Xsort)
  i=-2
  while(uLeng<k&&Xsort[n-k-i]>Xmean){
    i=i+1
    uLeng<-length(ufilt(X,u=Xsort[n-k-i],r))
  }
  if(uLeng>=k){
    return((Xsort[n-k-i]+Xsort[n-k-i-1])/2)
  }else{
    return(umax(X,k-1,r))
  }
}
```

Bibliography

Aloui, C. and Mabrouk, S. (2010). Value-at-risk estimations of energy commodities via long-memory, asymmetry and fat-tailed garch models. *Energy Policy*.

Atchadé, Y., Fort, G., Moulines, E., and Priouret, P. (2011). Adaptive markove chain monte carlo: Theory and methods. In *Baysian Time Series Models*. Cambrudge Univeristy Press.

Box, G. E. P. and Muller, M. E. (1958). A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*.

Christoffersen, P. F. (1998). Evaluating interval forecasts. In *International Economic Review*, volume 39, pages 841–862. Economics Department of the University of Pennsylvania.

Coles, S. (2001). *An Introduction to Statistical Modeling of Extreme Values*. Springer.

Dahlen, K. E. (2010). Comparison of acer and pot methods for estimation of extreme values. Master's thesis, Norwegian University of Science and Technology.

Dahlen, K. E., Solibakke, P. B., Westgaard, S., and Næss, A. (2015). On the estimation of extreme values for risk assessment and management: The acer method. *International Journal of Business*.

Eddelbuettel, D., Francois, R., Allaire, J., Ushey, K., Kou, Q., Bates, D., and Chambers, J. (2016). *Rcpp: Seamless R and C++ Integration*.

Fernandez, C. and Steel, M. F. T. (1998). On bayesian modeling of fat tails and skewness. *Journal of the American Statistical Assosiation*, 93(441).

Gamerman, D. and Lopes, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Champman and Hall/CRC, 2nd edition.

Gelman, A., Roberts, G. O., and Gilks, W. R. (1996). Efficient metropolis jumping rules. In *Bayesian Statistics 5*.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Giot, P. and Laurent, S. (2003). Market risk in commodity markets: a var approach. In *Energy Economics*, volume 25, pages 435–457. Elsevier.

Givens, G. H. and Hoeting, J. A. (2013). *Computational Statistics*. John and Sons, 2nd edition.

Karpa, O. and Næss, A. (2012). Extreme value statistics of wind speed data by the acer method. *Journal of Wind Engineering and Industrial Aerodynamics*.

Kupiec, P. H. (1995). Techniques for verifying the accuracy of risk measurement models. *Journal of Derivatives*.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*.

Müller, P. (1991). A generic approach to posterior integration and gibbs sampling. Technical report, Department of Statistics, Purdue University.

Næss, A. and Gaidai, O. (2009). Estimation of extreme value from sampled time series. In *Structural Safety*, volume 31, pages 325–334.

Næss, A., Gaidai, O., and Karpa, O. (2013). Estimation of extreme value by the average conditional exceedance rate method. *Journal of Probability and Statistics*.

Roberts, G. O. and Rosenthal, J. S. (2007). Coupling and ergodicity of adaptive markov chain monte carlo algorithms. *Journal of Applied Probability*.

Rødvei, K. K. (2015). Acer r packaged. unpublished.

Steen, M., Westgaard, S., and Gjølberg, O. (2015). Commodity value-at-risk modeling:comparing riskmetrics,historic simulation and quantile regression. *Journal of Risk Model Validation*.

Wuertz, D., Chalabi, Y., Miklovic, M., Boudt, C., and Chausse, P. (2013). *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling*.